MASTER EN SCIENCES INFORMATIQUES

Analyse et mise en place d'outils permettant l'exploitation des données Netflow

Luyckx, Christophe

Award date:
2015

Awarding institution:
Universite de Namur

Faculté d'informatique
Année académique 2014 - 2015

Analyse et mise en place d'outils permettant l'exploitation des données NetFlow

par Christophe Luycx

Promoteur
Laurent SCHUMACHER
Promoteur externe
Alain NINANE

Mémoire présenté en vue de l'obtention du grade de
MASTER EN INFORMATIQUE

Résumé

Ce mémoire a pour sujet l'analyse et la mise en place d'outils permettant l'exploitation des données NetFlow. L'objectif de celui-ci est de montrer que cette technologie permet d'appliquer la politique de sécurité réseau. Il aborde les aspects légaux relatifs à la capture et l'exploitation des NetFlow. Le protocole NetFlow et des recommandations pour la mise en place d'outils d'analyse y sont étudiés.

Un outil est choisi à l'aide d'une méthodologie et mis en place. Il est ensuite adapté à nos besoins.

Des scénarios sont étudiés afin de mettre en pratique notre étude et tester nos outils.

Abstract

This final dissertation has about the analysis and development of tools for the exploitation of NetFlow data. The purpose of it is to show that this technology allows to apply the network security policy. It discusses the legal aspects related to the capture and exploitation of NetFlow. The NetFlow protocol and recommendations for the development of analysis tools are studied there.

A tool is chosen using a methodology and set up. It is then adapted to our needs.

Scenarios are studied in order to practice our study and test our tools.

Remerciements

Il est impossible de rédiger un mémoire sans l'aide de plusieurs personnes. La première personne que je souhaite remercier est mon promoteur, Monsieur Laurent Schumacher qui m'a aidé, aiguillé et conseillé tout au long de l'élaboration de ce mémoire.

Je remercie également Monsieur Alain Ninane, responsable sécurité de l'UCL qui était mon promoteur externe. Il m'a aidé à trouver un sujet en rapport avec la sécurité et m'a ensuite guidé tout au long de mon mémoire.

Monsieur Olivier Bonaventure, professeur d'informatique dans le département d'ingénierie informatique de l'UCL, qui m'a également aidé à définir le sujet et Ramin Sadre, professeur dans le département d'ingénierie informatique de l'UCL qui a apporté des réponses à certaines interrogations.

Je souhaite également remercier l'UCL dans son ensemble pour m'avoir permis de réaliser ce mémoire en son sein.

Merci à Monsieur Quentin Hunin et Monsieur Dominique Margot Ingénieurs des infrastructures des réseaux du système d'information pour leur aide.

Monsieur Luca Deri pour son support concernant le produit Ntopng et le support de NetFlow Analyzer pour leur aide.

Un tout grand merci à ma compagne, Emilie Delferrière pour son soutien et sa patience.

Table des matières

Liste des tableaux

Table des figures

1 Introduction

Les cyberattaques ne cessent de s'intensifier. Très régulièrement, les médias en relatent. Personne ne semble épargné. La presse, les télévisions, les gouvernements, des professionnels de l'informatique, etc... sont victimes de cyberattaques d'origines diverses.

On constate que la plupart du temps les informaticiens n'ont pas pu anticiper ces attaques, qui font donc la une, le buzz. Les attaques qui exploitent des failles zero-day sont difficilement anticipables mais beaucoup d'attaques exploitent le non respect des bonnes pratiques. A contrario, de nombreuses attaques sont tuées dans l'œuf.

La sécurité informatique et plus particulièrement celle des réseaux est plus que jamais une priorité absolue. Le nombre de périphériques connectés ne cesse de croitre. Les utilisateurs veulent pouvoir se connecter à tout, de n'importe où et avec n'importe quel périphérique. On constate que la sécurité ne se développe pas au même rythme que les périphériques connectés. L'aspect sécurité est souvent laissé de côté. Les entreprises n'ayant pas été confrontées à une attaque ou ne s'en étant pas rendues compte ne voient pas l'utilité de mobiliser des ressources dans ce domaine.

Que faut-il mettre en œuvre pour appliquer la politique de sécurité réseau d'une organisation donnée ? La question se pose, à nous, dans ce travail, dans un environnement Cisco / NetFlow car notre organisation, à savoir l'UCL, est équipé de matériel Cisco.

Renseignements pris il n'y a pas à proprement parler de politique de sécurité dans notre contexte mais des codes d'éthique et de déontologie. Ces codes ont été mis en place par l'Université, afin que chacun sache ce qui peut se faire ou non pour préserver l'outil informatique et pour éviter les abus et infractions.

L'analyse et la mise en place d'outils permettant l'exploitation des données NetFlow permettent-elles de d'appliquer la politique de sécurité réseau de l'UCL, qui découle de ses codes d'éthique et de déontologie ?

Afin de répondre à cette question nous appliquerons la méthodologie suivante :

- Exploration des NetFlow : nous parcourrons la documentation Cisco pour étudier et comprendre le protocole NetFlow. Nous nous baserons sur un article écrit par l'Academic Network de Serbie pour présenter des recommandations concernant la mise en place de systèmes d'analyse NetFlow ;

- Sélection COTS : nous comparerons plusieurs outils permettant d'analyser les traces NetFlow et une méthode basée sur COTS permettra d'en choisir un en fonction de critères pertinents pour notre cas ;

- Déploiement et adaptation de la solution choisie : le produit choisi sera déployé et mis en production. Il sera personnalisé pour répondre au mieux à nos besoins ;

- Tests de situation : des scénarios seront étudiés afin de mettre en pratique notre étude et tester nos outils.

Le présent mémoire a plusieurs finalités : l'obtention du diplôme de maître en informatique à horaire décalé à l'UNamur, l'obtention du certificat InfoSafe organisée par l'UNamur et l'ICHEC (Certificat en management de la sécurité des systèmes d'information) et la mise en place des outils d'analyse réseau afin d'améliorer le niveau de surveillance du réseau de l'UCL. Le mémoire est donc destiné à deux publics. La version destinée à la formation InfoSafe se verra raccourcie. Les parties (2.6, 2,7 et 3.1) seront retirées et une partie relative au processus de définition de la politique de sécurité sera ajoutée.

2 Théorie

2.1 Environnement de travail

Le cadre environnemental de ce présent mémoire est l'Université Catholique de Louvain. Celle-ci est implantée sur six sites (Louvain-la-Neuve, Woluwe, Mons, Tournai, Saint-Gilles, Charleroi) et est composée de 14 facultés. L'UCL accueille en moyenne environ 30000 étudiants, 3600 chercheurs, 2000 doctorants, et supervise 1000 conventions de recherche. L'Université compte 5000 collaborateurs qui exercent 200 métiers différents. Nous pouvons constater que les utilisateurs du système d'information sont nombreux et variés.

Comme mentionné ci-dessus, nous sommes en présence de six sites distincts. Dans la présente étude, nous nous occuperons uniquement des sites de Louvain-la-Neuve et Woluwe. Ces deux sites sont connectés au niveau du réseau via un trunk 1 Gbps et 2 x 100 Mbps (liaison backup). Leur point de sortie vers l'extérieur est le même. Ce sont les flux NetFlow générés par ce routeur qui seront récupérés et analysés dans le cadre de ce mémoire.

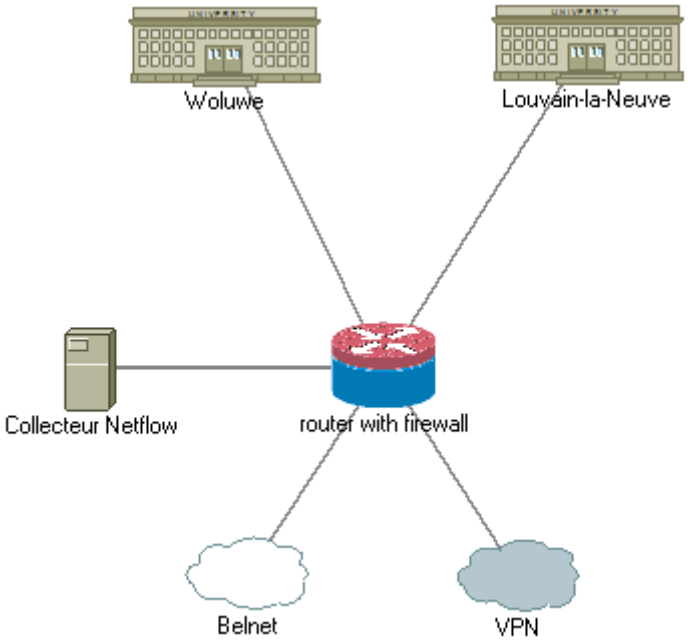


FIGURE 1 – Réseau simplifié de l'UCL

2.2 Aspects légaux

Il n'est pas possible de collecter des informations concernant le trafic réseaux en ignorant les aspects légaux.

Qu'en est-il de la loi du 8 décembre 1992 relative à la protection de la vie privée à l'égard des traitements de données à caractère personnel et d'autres normes notamment celles régissant le secret de la correspondance ?

Ce point concerne cette problématique. Les différents textes de loi sont appliqués au cas d'espèce.

a) Loi du 8 décembre 1992 relative à la protection de la vie privée à l'égard des traitements de données à caractère personnel [M.B., 18 mars 1993] [11]

Chaque entreprise agissant dans son secteur d'activités qu'il soit dans le domaine public ou privé, traite des données via son système informatique. Or, les informations traitées sont, à l'heure actuelle, importantes à préserver. Il y a donc lieu de les protéger. Auparavant, il était aisé de sécuriser le réseau informatique encore peu développé en fermant à clé un local, en ayant un programme performant ou encore en faisant un back up régulier. Les choses ont changé vu les nouvelles technologies de l'information. Le réseau s'est étendu et exige maintenant un système de protection plus complexe.

Les informations relatives aux flux IP collectées par ce système de surveillance des réseaux sont-elles des données à caractère personnel telles que définies par la législation vie privée ?

La présente loi définit les données à caractère personnel comme étant *« toute information concernant une personne physique identifiée ou identifiable, désignée ci-après ?personne concernée ? ; est réputée identifiable une personne qui peut être identifiée, directement ou indirectement, notamment par référence à un numéro d'identification ou à un ou plusieurs éléments spécifiques, propres à son identité physique, physiologique, psychique, économique, culturelle ou sociale »* [Article 1, loi du 8 décembre 1992] [11].

Les données récoltées sur le lieu de travail et relatives à l'identification du poste, les flux sortant, les protocoles utilisés, la destination permettent-elles d'identifier une personne physique identifiée ou identifiable ? La question du statut de l'adresse IP face aux données à caractère personnel fait débat. En effet, une adresse IP seule n'est pas en soi une donnée personnelle mais associée à d'autres informations, elle pourrait permettre l'identification d'une personne. Cependant, l'adresse IP permet surtout d'identifier une machine mais ne permet pas précisément d'en connaître un utilisateur. Afin d'accéder au réseau informatique de l'entreprise, un login et un mot de passe sont souvent requis pour un utilisateur précis. L'administrateur de ce réseau connaît donc le nom de la personne à qui est déféré tel mot de passe et tel login. Comment être certain que l'utilisateur n'a pas communiqué ses données de connexion à une autre personne ? Néanmoins, la Cour de justice de l'Union européenne a jugé, dans une décision du 24 novembre 2011 [Scarlet Extended SA contre société belge des auteurs, compositeurs et éditeurs SCRL (SABAM)] [14], que les adresses IP sont des données protégées à caractère personnel.

La législation relative à la vie privée s'appliquerait. Dès lors, les conditions de l'article 4 et suivant de Loi doivent être respectées, à savoir que ces données :

- doivent être traitées loyalement et licitement ;
- collectées pour des finalités déterminées, explicites et légitimes ;
- soient adéquates, pertinentes et non excessives au regard des finalités pour lesquelles elles sont obtenues et pour lesquelles elles sont traitées ultérieurement ;
- soient exactes et, si nécessaire, mises à jour ; toutes les mesures raisonnables doivent être prises pour que les données inexactes ou incomplètes, au regard des finalités pour lesquelles elles sont obtenues ou pour lesquelles elles sont traitées ultérieurement, soient effacées ou rectifiées ;
- soient conservées sous une forme permettant l'identification des personnes concernées pendant une durée n'excédant pas celle nécessaire à la réalisation des finalités pour lesquelles elles sont obtenues ou pour lesquelles elles sont traitées ultérieurement.

En outre, *« le traitement de données à caractère personnel ne peut être effectué que dans l'un des cas suivants :*

a) lorsque la personne concernée a indubitablement donné son consentement ;

b) lorsqu'il est nécessaire à l'exécution d'un contrat auquel la personne concernée est partie ou à l'exécution de mesures précontractuelles prises à la demande de celle-ci ;

c) lorsqu'il est nécessaire au respect d'une obligation à laquelle le responsable du traitement est soumis par ou en vertu d'une loi, d'un décret ou d'une ordonnance ;

d) lorsqu'il est nécessaire à la sauvegarde de l'intérêt vital de la personne concernée ;

e) lorsqu'il est nécessaire à l'exécution d'une mission d'intérêt public ou relevant de l'exercice de l'autorité publique, dont est investi le responsable du traitement ou le tiers auquel les données sont communiquées ;

f) lorsqu'il est nécessaire à la réalisation de l'intérêt légitime poursuivi par le responsable du traitement ou par le tiers auquel les données sont communiquées, à condition que ne prévalent pas l'intérêt ou les droits et libertés fondamentaux de la personne concernée qui peut prétendre à une protection au titre de la présente loi » [Article 5, loi du 8 décembre 1992] [11].

Il faut également souligner que l'article 16§4 de cette même loi impose à l'employeur, responsable du traitement des données, de *« prendre les mesures techniques et organisationnelles requises pour protéger les données à caractère personnel contre la destruction accidentelle ou non autorisée, contre la perte accidentelle ainsi que contre la modification, l'accès et tout autre traitement non autorisé de données à caractère personnel. Ces mesures doivent assurer un niveau de protection adéquat, compte tenu, d'une part, de l'état de la technique en la matière et des frais qu'entraîne l'application de ces mesures et,*

d'autre part, de la nature des données à protéger et des risques potentiels ». L'organisme doit mettre en place des mécanismes de journalisation et de traçage au niveau des accès internes mais aussi des accès externes de ses zones informatiques.

Nous sommes donc confrontés à deux intérêts contradictoires : d'une part, l'employeur a le droit de contrôler l'utilisation de l'outil informatique qu'il met à disposition et d'autre part, le travailleur a droit au respect de sa vie privée.

b) La convention collective du travail (CCT) n°81 du 26 avril 2002 relative à la protection de la vie privée des travailleurs à l'égard du contrôle des données de communication électroniques en réseau [12].

Le champ d'application de cette CCT se limite au secteur privé sans toutefois exclure le secteur public puisque la Commission vie privée estime que ce document est également un point de référence pour ce dernier.

L'employeur est tenu de respecter certains principes de base relatifs à la protection de la vie privée du travailleur lorsqu'il agit dans le cadre de la cybersurveillance :

- le principe de finalité : l'employeur agit dans un but légitime par exemple vérifier le bon fonctionnement du réseau et assurer la bonne exécution d'un service de communications électroniques (comme par exemple, le contrôle est autorisé s'il y a un téléchargement de gros fichiers qui ralenti ou paralyse des communications professionnelles, un téléchargement susceptible de contenir un virus, un transfert de courriers volumineux) ;

- le principe de proportionnalité : l'employeur doit s'en tenir au strict nécessaire afin de réaliser sa mission. Il y donc des contrôles ponctuels et justifiés par des indices suspectant une utilisation abusive des outils de travail ;

- le principe de transparence : l'employeur doit informer le travailleur des mesures prises dans le cadre du contrôle électronique qu'il effectue soit via son règlement de travail, soit via le contrat de travail individuel, soit via une convention collective du travail [Commission vie privée, loi vie privée] [8].

L'article 5 de cette CCT affirme explicitement le principe de finalité et prévoit que *« le contrôle de données de communication électroniques en réseau n'est autorisé que lorsque l'une ou plusieurs des finalités suivantes est ou sont poursuivies :*

1° la prévention de faits illicites ou diffamatoires, de faits contraires aux bonnes moeurs ou susceptibles de porter atteinte à la dignité d'autrui ;

2° la protection des intérêts économiques, commerciaux et financiers de l'entreprise auxquels est attaché un caractère de confidentialité ainsi que la lutte contre les pratiques contraires ;

3° la sécurité et/ou le bon fonctionnement technique des systèmes informatiques en réseau de l'entreprise, en ce compris le contrôle des coûts y afférents, ainsi que la protection physique des installations de l'entreprise ;

4° le respect de bonne foi des principes et règles d'utilisation des technologies en réseau fixés dans l'entreprise ».

L'employeur peut donc consulter les données relatives aux sites internet telles que la durée de connexion, les adresses des sites consultés par ordinateur et les données relatives aux courriers électroniques comme, par exemple, le nombre et le volume de courriers sortants/entrants.

c) Loi du 13 juin 2005 relative aux communications électroniques [M.B., 20 juin 2005] [13]

Dans un premier temps, l'article 124 de la loi du 13 juin 2005 concernant les communications électroniques interdit à toute personne n'ayant pas été autorisée de prendre connaissance de l'existence d'une information transmise par voie électronique qui ne lui est pas destinée personnellement ou d'identifier les personnes concernées par cette transmission d'informations ainsi que son contenu ou de prendre connaissance de données en matière de communications électroniques relatives à une autre personne ou encore de modifier, supprimer révéler, stocker ou faire usage de l'information, de l'identification ou des données obtenues.

En outre, les articles 259bis § 1 du code pénal concernant les écoutes, la prise de connaissance et de l'enregistrement de communications de télécommunications privées et 314bis du code pénal concernant les infractions au secret des communications et des télécommunications privées condamnent cette pratique.

Dans un second temps, l'article 125 de la loi du 13 juin 2005 concernant les communications électroniques établit des exceptions aux articles cités ci-dessus en fonction de la finalité de la collecte d'informations et notamment *« lorsque les actes visés sont accomplis dans le but exclusif de vérifier le bon fonctionnement du réseau et d'assurer la bonne exécution d'un service de communications électroniques ».*

En d'autres termes, la collecte d'informations, dans le cas présent, a comme finalité de vérifier le bon fonctionnement du réseau et la bonne exécution des services. Cette finalité est reprise dans les exceptions de l'article 125 § 1er de la loi du 13 juin 2005, rendant l'article 124 et les articles 259bis et 314bis non applicables. La collecte est donc tout à fait licite.

S'il y a une anomalie détectée au niveau de la sécurité et au niveau du bon fonctionnement du système, l'employeur peut effectuer un contrôle pour identifier l'auteur.

d) Conclusion

Il faut toujours faire une balance des intérêts et établir un équilibre entre le respect de la vie privée des travailleurs et les intérêts professionnels des employeurs. Dans le cas présent et au vu de l'ensemble de ces législations, la collecte d'informations concernant le trafic réseaux qui a pour but de vérifier le bon fonctionnement du réseau et la bonne exécution d'un service de communications électroniques, est licite, proportionnée et transparente.

2.3 Netflow

2.3.1 La problématique

Les réseaux IP sont devenus de plus en plus importants au fil des années. Un besoin d'exercer un contrôle des réseaux est né et n'a cessé de croître. Pour cela, SNMP est encore largement utilisé aujourd'hui pour surveiller les éléments actifs mais se limite à la couche 7 du modèle OSI. Pour des incidents ponctuels, un analyseur de traces tel que Wireshark peut aussi être utilisé. Les besoins vont grandissants et les moyens financiers mis à disposition pour traiter cette problématique sont souvent limités. Une solution répondant à ce besoin et à moindre coût a dû être trouvée (Clément Lauréat [2012] [25]).

2.3.2 La solution

Il a fallu trouver une solution au problème tout en respectant les limites budgétaires fixées. Des technologies permettant l'analyse des réseaux IP ont été développées. Elles portent des noms différents en fonction du constructeur qui les implémente mais leurs fonctionnalités sont équivalentes. Sans les citer toutes en voici quelques unes : NetFlow de Cisco, Jflow de Juniper Networks, NetStream de Huawei Technologies, sFlow plusieurs vendeurs,...(NetFlow - en.wikipedia.org [1])

Ces technologies collectent le trafic IP qui entre ou sort d'une interface. Elles permettent par exemple de connaître la source et la destination du trafic, le protocole utilisé. L'équipement réseau de l'UCL étant de marque CISCO, nous nous occuperons donc du protocole développé par la marque à savoir NetFlow.

2.3.3 Le protocole NetFlow

Cette section est consacrée à l'étude du protocole NetFlow, elle est basée sur le guide des services NetFlow développé par Cisco [2007] [17], NetFlow - fr.wikipedia.org [4] et NetFlow - en.wikipedia.org [2].

NetFlow est un protocole propriétaire introduit par Cisco dans ses équipements réseaux dans le but de collecter des informations sur le trafic réseau et d'aussi en assurer la surveillance.

NetFlow peut avoir plusieurs usages :

Surveillance réseau : NetFlow permet de surveiller le réseau quasiment en temps réel. L'analyse des flux peut être utilisée pour visualiser le trafic lié à chaque routeur ou commutateur du réseau ou le visualiser d'une façon globale. Cette surveillance permet de detecter et de résoudre rapidement les problèmes rencontrés.

Surveillance des applications et profiling : NetFlow permet d'obtenir une vue sur l'utilisation des applications sur le réseau. Cela aide à allouer aux utilisateurs les ressources réseaux et applicatives nécessaires et de comprendre les nouveaux services

Surveillance des utilisateurs et profiling : NetFlow permet d'obtenir l'utilisation du réseau et des ressources par les utilisateurs et les applications. Ce qui permet de planifier l'utilisation des ressources et du réseau et de détecter des problèmes de sécurité ou l'utilisation de certaines ressources illicites.

Planification du réseau : La capture des informations sur une longue période permet d'anticiper la croissance du réseau et de planifier des mises à jour au niveau des équipements réseaux. NetFlow permet d'optimiser la planification des mises à jour. Il permet aussi d'optimiser l'utilisation du réseau, sa performance, sa fiabilité et ainsi de diminuer les coûts d'utilisation. NetFlow permet aussi de détecter du trafic réseau non désirable, de valider des QoS et d'analyser de nouvelles applications réseau.

Analyse de sécurité : NetFlow permet d'identifier des attaques de déni de service, des virus et vers en temps réel. Il permet de mettre en évidence un changement de comportement dans le réseau, ce qui indique souvent une anomalie. Les données peuvent aussi être utilisées pour la prévention, pour permettre de comprendre des attaques et aussi de les rejouer.

Accounting et facturation : NetFlow fournit des informations facilitant la facturation (IP, type de service ou d'application, temps d'utilisation d'un service ou d'une application, utilisation de la bande passante, qualité de service)

Marketing : NetFlow peut être utilisé à des fins marketing en analysant l'utilisation du réseau et en répondant aux questions : qui ?, quand ?, où ?, combien de temps ?

Le mémoire est axé sur l'analyse des NetFlow dans le but de mettre en évidence les comportements anormaux sur le réseau, afin de détecter des attaques ou des usages illicites. Nous nous occuperons donc principalement de l'aspect analyse de sécurité, de la surveillance réseau et des usages.

Définition d'un flux réseau NetFlow

La technologie NetFlow s'appuie sur la notion de flux. Un flux est un échange de packets unidirectionnel entre une source et une destination. Il est caractérisé par 7 champs clés : une adresse IP source et une adresse IP destination (couche réseau), un port source (pour les protocoles UDP ou TCP, 0 pour les autres protocoles) et destination (couche transport), le protocole de couche 3, Type of Service, l'interface en entrée.

Les 7 champs définissent un flux de façon unique. Pour qu'un flux soit considéré comme un nouveau flux, un de ces 7 champs doit être différent. Dans le cas contraire il sera décompté des statistiques. D'autres champs secondaires existent (ex : date et heure du début du flux).

Cache NetFlow

NetFlow utilise un cache qui contient des informations pour tous les flux actifs. Le cache NetFlow est construit à partir du premier paquet d'un flux. Un enregistrement pour chaque flux actif est maintenu dans le cache. Chaque enregistrement contient des champs clés qui pourront être utilisés pour exporter des données. Chaque fois qu'un paquet est reçu, le cache est mis à jour. Soit le flux existe déjà dans le cache et dans ce cas les compteurs sont incrémentés, soit le flux n'existe pas encore et une nouvelle entrée est créée dans le cache. Le cache est exporté à intervalles réguliers (souvent 5 minutes) vers un collecteur de flux. Ce collecteur contient l'historique et les informations de tous les flux qui ont été acheminés à travers le routeur ou le switch. NetFlow enregistre des informations sur chaque paquet, ce qui permet de fournir une vue très détaillée du traffic réseau. Le volume des données collectées peut représenter 1,5% du trafic routé.

La gestion du cache est très performante, elle permet de gérer un grand nombre de flux concurrents et de courtes durées. La gestion du cache NetFlow est composée d'algorithmes sophistiqués, qui permettent de déterminer si un paquet appartient à un flux existant ou si une nouvelle entrée doit être créée dans le cache. Il permet aussi de mettre à jour de façon dynamique les mesures concernant un flux contenu dans le cache et de déterminer l'expiration d'un flux.

Les règles d'expiration d'un flux dans le cache :

- Les flux qui sont inactifs pendant un temps déterminé (15 sec par défaut) expirent et sont retirés du cache.
- Les flux actifs trop longtemps (30 minutes par défaut) expirent et sont retirés du cache.
- Si il s'agit d'un flux TCP et que le flag FIN ou RST est détecté, le flux expire et est retiré du cache.

Les flux qui ont expiré sont regroupés dans une trame d'exportation NetFlow pour être exportés. Pour configurer l'exportation, il suffit de spécifier l'adresse IP et le port du collecteur NetFlow. Un collecteur NetFlow est un périphérique qui permet d'analyser et filtrer les trames NetFlow exportées.

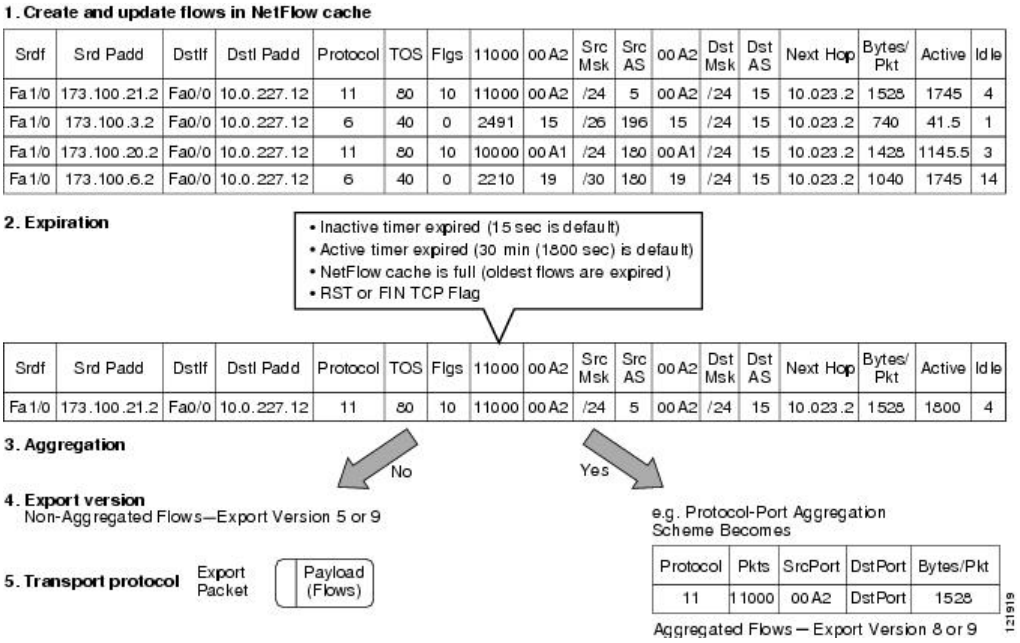


FIGURE 2 – Cache NetFlow [Cisco.com]

Les différentes versions du format d'exportation NetFlow

La trame d'exportation contient une entête et une série d'enregistrements de flux. L'entête contient des informations comme le numéro de séquence, le numéro d'enregistrement ou le sysuptime. Les enregistrements de flux contiennent des informations sur le flux comme l'adresse IP, le port ou des informations de routage.

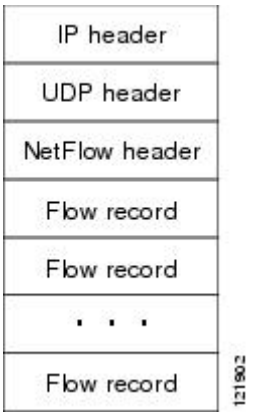


FIGURE 3 – Trame d'exportation NetFlow pour les versions 1, 5, 7, 8 [Cisco.com]

La version 1 est la version initiale, elle est à présent obsolète (Limitée à IPv4 sans masque réseau ni numéro de système autonome). Les versions de 2 à 4 n'ont jamais été publiées et ne sont donc pas supportées. La version 5 est la version sans doute la plus répandue, elle ajoute le protocole BGP. Cette version est limitée à l'IPv4. La version 6 n'est plus prise en charge par Cisco. La version 7 est semblable à la version 5 mais avec un champ "router source" en plus.

La plus récente est la 9, elle est basée sur des templates. Les templates offrent plus de flexibilité et autorisent l'ajout de champs sans redéfinir le standard, ce qui permet de suivre des flux IPv6, IPv4, multicast ou MPLS. Les développeurs d'applications pour NetFlow (collecteur, visualisation,...), ne doivent plus recompiler à chaque fois qu'un nouveau champ est rajouté. Ils peuvent utiliser un fichier qui documente le format du template. Des nouvelles fonctionnalités peuvent être rajoutées plus facilement et donc plus rapidement sans devoir modifier les implémentations existantes. La version 9 est prête pour le futur car elle peut être adaptée pour supporter de nouveaux protocoles.

IPFIX (Internet Protocol Flow Information Export) qui est basé sur la version 9 de NetFlow est un protocole IETF. Il a été créé par la nécessité d'avoir un format standard d'exportation flux pour les routeurs. Les spécifications pour IPFIX sont documentés dans les RFC 7011 à 7015 et RFC 5103.

2.4 Les alternatives aux NetFlow

La technologie abordée ici est l'analyse des flux NetFlow mais nous allons la comparer à d'autres technologies. Nous allons tenter de dégager les avantages et inconvénients et pour conclure nous déterminerons si le choix de NetFlow était approprié ou non.

2.4.1 NetFlow *versus* Système de détection d'intrusion réseau (ou NIDS : Network Intrusion Detection System)

2.4.1.1 Présentation

Le NIDS analyse les paquets en transit sur le réseau afin de repérer les activités anormales ou suspectes. Une alarme est générée lorsqu'une anomalie est suspectée. Nous employons le terme "suspectée" car des faux positifs peuvent être détectés, c'est à dire qu'une activité normale est considérée comme anormale par le système.

Il y a principalement deux types de détection.

Le premier est basé sur une base de données de signature d'attaques connues. Lorsqu'une correspondance est trouvée une alarme est émise. Ce type constitue un IDS traditionnel.

La deuxième est basée sur des anomalies (forte ressemblance avec le pattern matching). Dans un premier temps un modèle statistique décrivant le trafic réseau normal est construit. Quand un comportement s'écarte significativement du modèle une alerte est levée. Une période d'apprentissage et une configuration minutieuse sont donc nécessaires.

Le deuxième type a été ajouté pour régler les grosses lacunes du premier. Il est basé sur l'analyse des enregistrements NetFlow. Notre comparaison reviendra donc à comparer les deux types de détection.

Les systèmes NIDS requièrent du matériel dédié disposant d'interface(s) réseau(x) en mode furtif (pas d'adresse IP).
La présentation a été faite selon l'article de Earl Carter [2002] [16] et le livre de Roberto Di Pietro et Luigi Mancini [2008] [27].

2.4.1.2 Comparaison

Avec les traces NetFlow, nous n'avons pas d'informations concernant le contenu des paquets. C'est une des grosses différences. Les flux ne contiennent pas d'informations des couches supérieures du modèle OSI, ils contiennent juste le profil du traffic. Les NetFlow ne permettent donc pas d'analyser profondément les paquets.

N'effectuant pas d'analyse des paquets, l'analyse NetFlow est donc plus rapide. Ce point peut-être intéressant pour les réseaux chargés.

L'analyse des NetFlow peut-être très efficace pour détecter des attaques de failles zero-day ou des attaques "mutantes". Un système de détection basé sur des signatures, aura du mal à les détecter.

Les NetFlow venant directement des routeurs, il est très facile d'obtenir une vue unique de l'entièreté du réseau.

Si l'analyse NetFlow est effectuée minutieusement, les vers ou autres anomalies réseaux peuvent être détectés très tôt.

L'analyse des NetFlow est quasi gratuite (si on ne prend pas en compte l'achat du matériel Cisco) car c'est une simple option à activer sur les routeurs et ne nécessite donc pas de matériel supplémentaire. Dans notre cas le matériel est déjà en place.

2.4.2 NetFlow *versus* SNMP (Simple Network Management Protocol

2.4.2.1 Présentation

SNMP est un protocole implémenté au niveau de la couche application du modèle OSI permettant d'obtenir des informations provenant de plusieurs systèmes différents interconnectés.

Sur chaque système pour lequel on veut obtenir des informations, un agent SNMP doit être présent. Les agents font la plupart du travail et donc la configuration est un peu plus complexe que pour le manager. Les informations collectées sont stockées dans une base de données appelée MIB (Management Information Base)

Un composant central envoyant les requêtes SNMP vers les agents est nécessaire (SNMP manager) afin d'obtenir les informations de ceux-ci.

La présentation a été faite selon l'article de Justin Ellingwood [2014] [19] et le livre de Pierre-Alain Goupille [2008] [22].

2.4.2.2 Comparaison

SNMP est un protocole assez simple et facile à mettre en place. Au niveau du réseau, il nous permet de monitorer l'utilisation de la bande passante. Il permet donc de détecter une anomalie concernant l'utilisation de la bande passante au niveau de l'interface physique.

SNMP permet, contrairement à NetFlow, d'obtenir d'autres informations que l'utilisation de la bande passante. Dans notre cas, ces informations ne sont pas utiles.

Point négatif, la vision se limite au niveau de l'interface. Il y a un manque de granularité qui ne permet pas de savoir quelle adresse IP source / destination ou encore quel programme sont concernés. Les NetFlow apportent cette granularité

2.4.3 NetFlow *versus* Software-Defined Network (SDN) / OpenFlow

2.4.3.1 Présentation

SDN est une forme de virtualisation, il sépare la couche de données de la couche de contrôle de ces données. Les commutateurs ou routers ne servent donc plus que d'aiguillage. Les services réseaux eux sont rassemblés dans un contrôleur unique. L'avantage est donc de n'avoir qu'un seul point de gestion.

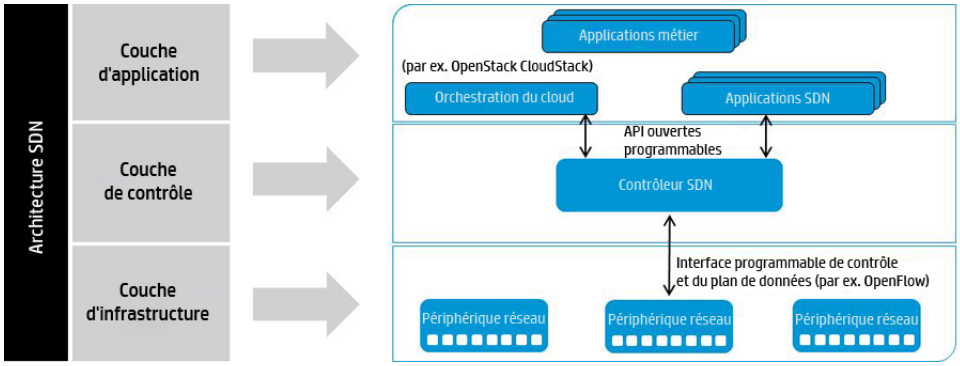


FIGURE 4 – Architecture SDN [HP.com]

Pour que des matériels de fabricants différents puissent recevoir des requêtes d'un contrôleur unique, un protocole et des interfaces de programmation sont nécessaires. L'Open Networking Foundation (ONF) a défini le protocole OpenFlow. Certains constructeurs l'adoptent (HP, IBM, Ericsson ou Alcatel-Lucent). Cisco, lui, offre une interface OpenFlow mais développe parallèlement sa propre solution, OnePK.

Dans un réseau classique, le routage est effectué par le commutateur ou le routeur en fonction des règles de routage configurées dans son système d'exploitation. Tous les paquets ayant la même destination suivent la même route. Les modèles haut de gamme permettent d'adapter la règle de routage en fonction du programme. Ces règles sont statiques et doivent être modifiées manuellement en cas de changement nécessaire.

SDN permet de rendre les règles de routage dynamiques et de les automatiser. Des règles peuvent être appliquées en fonction de l'heure ou de l'utilisation du réseau afin de donner plus ou moins de bande passante à certains services.

La présentation a été faite selon les articles de Yogita Hande, Aishwarya Jadhav, Achaleshwari Patil, Rutuja Zagade [2014] [23] et de Jean-Pierre Soulès [2014] [30].

2.4.3.2 Comparaison

SDN, séparant la partie matérielle de la partie logicielle, permet de rendre la gestion du trafic indépendante du fabriquant de matériel réseau. Ceux-ci sont relégués au deuxième rang. On peut donc commander un réseau hétérogène entier depuis un logiciel sans devoir

effectuer des modifications sur chaque équipement. Le réseau est par conséquent moins complexe et plus facile à gérer.

SDN peut être un outil puissant pour un système IDS car il permet d'obtenir un miroir du trafic réseau et de bloquer le trafic anormal dès qu'il est détecté. L'IDS peut avertir le contrôleur SDN qui modifie en temps réel les règles de routage afin d'isoler l'anomalie.

SDN / OpenFlow et NetFlow utilisent le même type de données (adresse IP source / destination, port source / destination, protocole, ...), mais l'objectif de leur utilisation est bien différente. Dans le cas de SDN, elles sont utilisées pour le routage des paquets mais NetFlow les utilise pour générer un audit du trafic réseau. Le fait que ces deux outils utilisent le même type de données et que le mot "flow" apparaisse dans leur appellation peut entraîner une confusion entre les deux.

La comparaison a été faite selon les articles de Yogita Hande, Aishwarya Jadhav, Achaleshwari Patil, Rutuja Zagade [2014] [23] et de Chris Smithee [2012] [29].

2.4.4 NetFlow : Le bon choix ?

Nous avons présenté différentes solutions permettant l'analyse du trafic réseau à des fins de sécurité.

La première solution NIDS, est basée sur l'analyse des paquets et une base de données de signatures. Cette solution est très efficace pour détecter des menaces connues mais inefficace face à des attaques ayant mutées ou de nouvelles menaces. Les NetFlow apportent une solution vis à vis de cette problématique.

SNMP permet de monitorer le réseau mais manque cruellement de granularité.

SDN est prometteur et offre une gestion centralisée et dynamique du réseau. Il amène des nouvelles notions et avec lui de nouveaux risques qui devront être analysés. Les logiciels de sécurité ne sont pas tous prêts pour cette technologie assez récente. Gartner nous met d'ailleurs en garde à ce sujet [20]. Avant d'être implémentée, cette technologie demandera donc une étude approfondie. Une refonte complète du réseau sera par ailleurs nécessaire.

Associer les fonctionnalités d'un ou plusieurs IDS à celles de SDN pourrait être une solution de sécurité performante si elle est étudiée et mise en place avec rigueur. SDN permet d'isoler rapidement une menace pour éviter qu'elle se propage ou n'affecte le reste du réseau et de façon automatique.

Pour résumer, on ne peut pas faire reposer entièrement la sécurité sur une seule solution. Un modèle en couche est fortement recommandé. Une solution comble les lacunes d'une autre. Si l'une tombe, on peut espérer que la seconde détectera la menace ou mieux la bloquera.

NetFlow est une solution efficace, facile à mettre en place car il suffit de l'activer et donc peu coûteuse (mise à part l'achat du matériel Cisco). NetFlow est donc un très bon choix et une bonne base pour améliorer la sécurité. Il nous permet déjà d'obtenir de nombreuses informations sur le fonctionnement du réseau et de détecter des anomalies.

Comme indiqué plus haut, un dispositif de sécurité en couche est la meilleure solution. On peut imaginer la mise en place d'un firewall de nouvelle génération du style "Palo Alto" en entrée et un analyseur NetFlow couplé à un IDS à l'intérieur de l'entreprise.

Il est important de noter que même si ces systèmes renforcent et améliorent la sécurité, aucun système ne sera jamais infaillible. Les systèmes de sécurité devront être en permanence améliorés.

	NetFlow	**NIDS**	**SNMP**	**SDN**
Détection Signature	V	V	X	-
Détection zero day	V	X	V	-
Granularité	forte	-	faible	-
Maturité technologique	forte	forte	forte	faible

TABLE 1 – Alternatives NetFlow

2.5 Solutions existantes au sein de l'UCL

Actuellement à l'UCL, les Netflow sont collectés sur un serveur et sont compressés et cryptés. Cette collecte s'effectue par intervalle de 5 minutes. Règle de nommage des logs : ft-v(version de Netflow).date.heure.xz.gpg

```
28 jan 13:40 ft-v07.2015-01-28.134000+0100.xz.gpg
28 jan 13:45 ft-v07.2015-01-28.134500+0100.xz.gpg
28 jan 13:50 ft-v07.2015-01-28.135000+0100.xz.gpg
28 jan 13:55 ft-v07.2015-01-28.135500+0100.xz.gpg
28 jan 14:00 ft-v07.2015-01-28.140000+0100.xz.gpg
28 jan 14:05 ft-v07.2015-01-28.140500+0100.xz.gpg
```

Code 1 – Stockage Netflow

Ces logs sont ensuite analysés via des scripts bash qui s'exécutent automatiquement à intervalles réguliers.

Les différentes fonctions de ces scripts permettent :

1) de trouver les adresses IP communiquant avec une adresse IP spécifique. Le but est d'identifier les machines faisant partie de botnet. Il est évident que la détection ne peut se faire que pour des botnets dont l'adresse du serveur est connue. De la même manière, nous pouvons identifier des postes infectés par des virus ou des vers connus. Pour ce faire, un fichier contenant des adresses IP connues est chargé par le script.

2) d'identifier les postes qui communiquent avec des pays "suspects" et inversement de repérer des connections ou tentatives de connection venant de pays n'étant pas censés le faire.

```
#! /bin/bash

  function clean
  {
   cat $1 | sed -e '/DstIPaddress/d' -e '/^$/d' | sort -o nftrack-$ipaddr.txt
    rm -f $1
  }

  . ~nfr/etc/nfw.conf

  if [ $ -ne 3 ]; then
    echo "Usage: $0 {host|from|to|...} ipAddress nMinutes"
    exit 1
  fi

  type=$1
  ipaddr=$2
  nmin=$3

  tmpout=.output.$$
  filters=.filters$$

  cat /dev/null > $tmpout
  cat ~/etc/filter-nftrack-template.cfg | sed -e s/IPADDR/$ipaddr/ > $filters
```

```
  trap "clean $tmpout ; rm -f $filters" EXIT

  for flow in $(find -L $NTFL_DATA -type f -mmin -$nmin -print | sort -n ); do
    ~/nftrack/11-nftrack-process.sh $filters $type $ipaddr $flow >> $tmpout
  done

exit
```

Code 2 – Exemple de script

Les outils graphiques qui seront mis en place n'ont pas comme objectif de remplacer ces scripts mais bien de les compléter. Leur avantage est de permettre une représentation des Netflow sous forme de graphes. Cette manière de représenter les données met en évidence certaines choses qu'il est parfois compliqué de voir via des scripts.
Nous pouvons ainsi visulaliser l'occupation de la bande passante, les parts occupées par chaque protocole, l'évolution de la charge réseau au cours d'une journée, afficher des statistiques, etc.. Les interfaces offertes par les différents outils graphiques simplifient et améliorent l'analyse des données. Etant complémentaires, ces outils permettent aussi de confirmer ou de lever le doute sur les observations obtenues via les scripts.

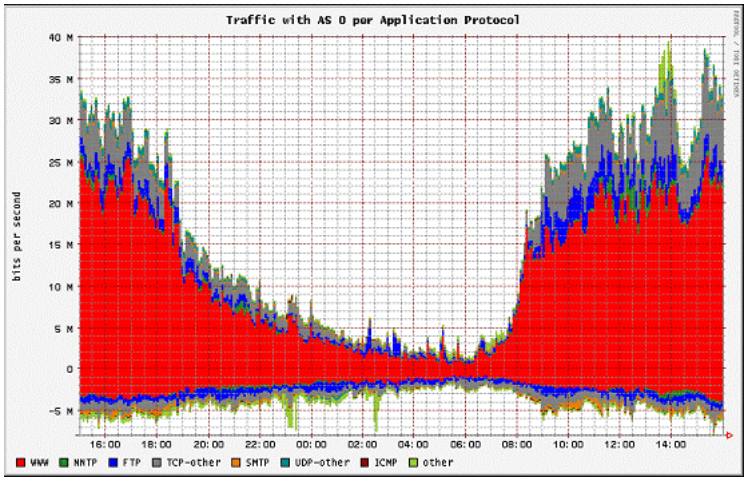


FIGURE 5 – Exemple de graphique

2.6 Recommandations

Nous allons présenter ici les recommandations développées par Ivan Ivanovic [2011] [24], pour la capture et l'analyse des NetFlow. Nous regarderons du côté de l'architecture, où placer les éléments d'analyse et de capture. Nous verrons aussi comment les configurer correctement. Nous pourrons également comparer ces recommandations à notre environnement de travail, afin de vérifier si elles sont respectées.

2.6.1 Architecture

La plupart du temps nous avons un ensemble de composants réseaux qui génèrent et exportent des données de trafic et des composants qui les analysent. Il peut y avoir plusieurs composants qui collectent les données de traffic réseau mais l'idéal est de n'avoir qu'un seul périphérique qui traite les données.

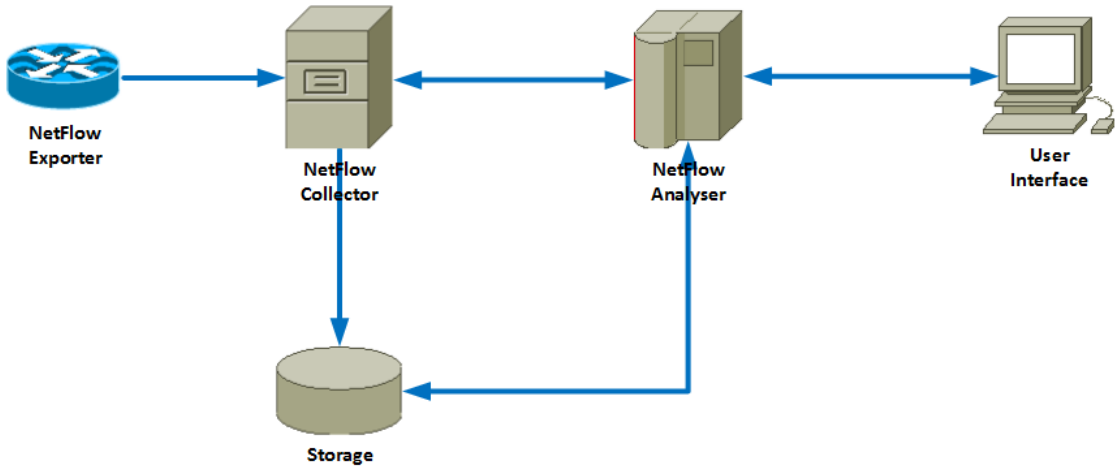


FIGURE 6 – Composants du système d'analyse

NetFlow Exporter : périphérique qui collecte les données relatives au traffic rentrant et sortant par ses interfaces et les exporte au format NetFlow (Version 5,7 ou 9) ;

NetFlow Collector : collecte les données venant du NetFlow Exporter et les compresse ;

NetFlow Analyser : traite les données collectées en fonction de critères définis par l'administrateur du système. Les résultats peuvent être stockés dans une base de données pour une réutilisation ultérieure ;

Storage : permet de stocker les NetFlow bruts et/ou les résultats des analyses effectuées sur ceux-ci ;

User Interface : une application, généralement web, qui permet de visualiser les résultats des analyses et de les paramétrer ;

Dans le schéma ci-dessus, nous pouvons constater que nous n'avons qu'un seul composant qui exporte des informations mais nous pourrions avoir plusieurs périphériques réseaux qui exportent des informations sur un serveur central.

2.6.1.1 Positionnement du collecteur dans le réseau

La position du collecteur dépend de l'architecture du réseau. La taille des données NetFlow exportées est proportionnelle à la densité du trafic qui passe par le routeur qui exporte ces données. Nous pouvons estimer que la taille du trafic NetFlow est environ 1% de la taille totale du trafic réseau. La distance entre le collecteur et le router (exporter), n'a donc pas beaucoup d'importance. L'accessibilité et la sécurité du collecteur sont des éléments beaucoup plus importants.

Généralement le collecteur est physiquement connecté au noeud principal. La plupart du traffic passant par celui-ci, c'est un choix logique. Dans notre cas, nous sommes effectivement connecté au router principal (traffic entrant / sortant du réseau UCL).

Recommandations concernant la sécurisation du collecteur :

- mettre le collecteur dans un VLAN séparé réservé à la gestion ;
- configurer un firewall sur le collecteur, afin de contrôler les accès ;
- s'assurer que le serveur NetFlow est disponible pour l'analyse du trafic même dans le cas où certains composants réseaux sont défectueux. Il est donc nécessaire d'avoir un UPS pour fournir de l'électricité en cas de panne au serveur.

2.6.1.2 Configuration de l'exportation des NetFlow

La configuration du l'exportation dépend des périphériques et de l'architecture du réseau. Sur les nouveaux périphériques, il est possible de régler l'exportation des NetFlow au niveau des interfaces en entrée ou en sortie. Certains modèles permettent de la faire dans les deux directions en même temps. Sur les modèles plus anciens, la capture ne se fait qu'en entrée.

Le problème principal dans la configuration de l'exportation NetFlow est la duplication des données NetFlow exportées. Ce problème peut être divisé en trois groupes.

2.6.1.3 Premier groupe : Uniquement le composant central supporte les NetFlow

Les enregistrements NetFlow sont exportés à partir de l'élément central. Ci-dessous, nous voyons une exportation NetFlow mal configurée et la correction du problème à la figure 8.

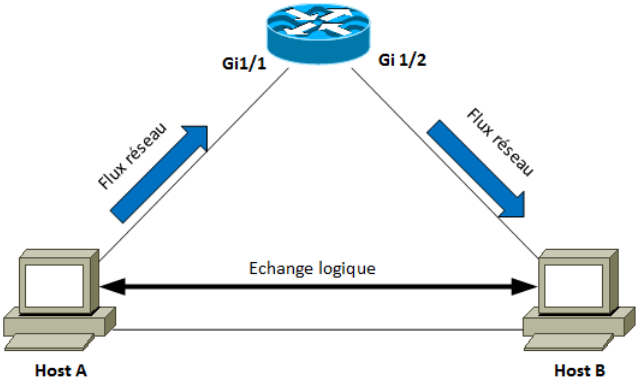


FIGURE 7 – Exportation NetFlow mal configurée

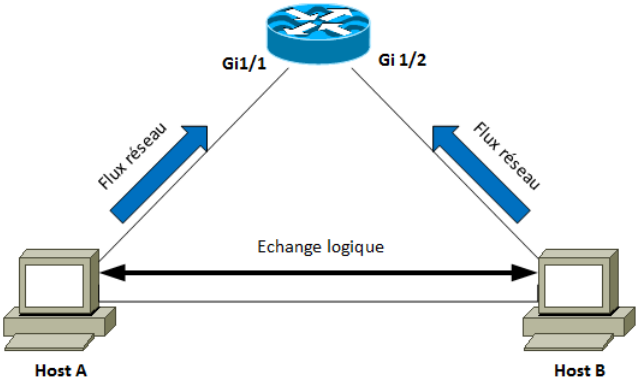


FIGURE 8 – Exportation NetFlow correctement configuré

La configuration de la figure 7 n'est pas correcte car nous aurons deux fois le flux entre Host A et Host B. Une première fois sur l'interface d'entrée Gi1/1 lorsque le flux rentre dans le routeur et une seconde fois sur l'interface de sortie Gi1/2 lorsque le flux sort du routeur vers le Host B. Non seulement l'information concernant la communication du Host A vers le Host B sera dupliquée mais l'information concernant la communication du Host B vers le Host A sera perdue.

Pour que l'information soit collectée correctement il faut qu'elle soit configurée sur toutes les interfaces soit en entrée soit en sortie.

La figure 8 montre une configuration correcte de l'exportation NetFlow. La collecte se fait sur toutes les interfaces uniquement en entrée. La collecte de l'information concernant la communication de Host A vers Host B se fera sur l'interface Gi1/1 et la collecte de l'information concernant la communication de Host B vers Host A se fera sur l'interface Gi1/2. Toute l'information passant par ce composant sera de cette manière correctement collectée et exportée.

Il se peut qu'il y ait certains subnets dans le réseau qui ne passent pas par l'élément central. Dans ce cas, les traces NetFlow concernant ces communications ne seront pas collectés. On suppose bien sûr que le périphérique par lequel passe la communication n'exporte pas lui même les NetFlow.

La figure 9 ci-dessous illustre ce cas de figure. Le router 1 est correctement configuré et exporte les informations concernant les communications passant à travers ses interfaces. Le router 2 n'est pas configuré pour exporter les enregistrements NetFlow ou ne le permet pas tout simplement. Les enregistrements NetFlow concernant les communications passant par ce routeur seront donc perdus.

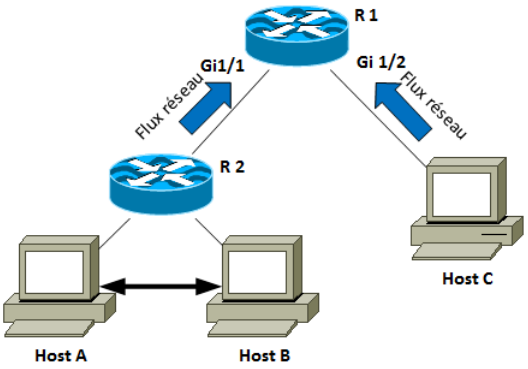


FIGURE 9 – Informations non collectées

2.6.1.4 Deuxième groupe : Les équipements frontaliers supportent les NetFlow

Ici nous sommes dans le cas où le réseau est divisé en plusieurs parties sur des sites différents qui sont connectés via un réseau MPLS d'un fournisseur d'accès. La configuration ressemble à la configuration de l'UCL mais les enregistrements NetFlow ne sont pas collectés sur chaque routeur frontalier comme dans l'exemple ici. La configuration et la collecte des enregistrements NetFlow doivent donc être configurées sur chaque routeur mais seulement sur les interfaces connectant la fin des différents sous-réseaux. La collecte peut être faite en entrée ou en sortie sur les interfaces mais le sens choisi doit être le même pour tout le réseau.

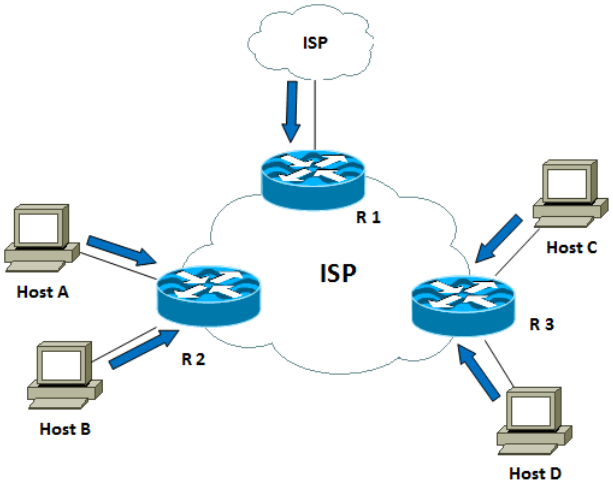


FIGURE 10 – Configuration collecte NetFlow sur les routeurs frontaliers

Les informations de communication entre Host A et ISP et entre tous les subnets sont collectées. Nous supposons pour cela que tous les composants réseaux supportent la technologie NetFlow et qu'ils sont correctement configurés.

2.6.1.5 Troisième groupe : Duplication du trafic

L'impossibilité d'éviter la duplication de NetFlow au niveau de la configuration hardware apparait parfois dans des réseaux complexes. La figure 11 illustre ce cas de figure.

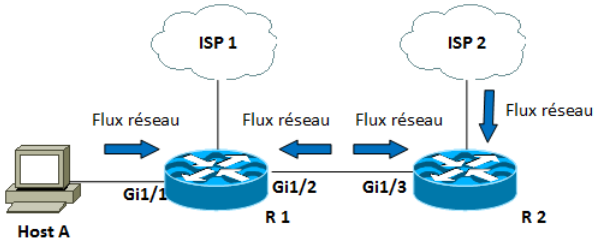


FIGURE 11 – Duplication de données NetFlow

Comme nous pouvons le voir, les routeurs 1 et 2 sont configurés de manière à collecter des NetFlow sur le trafic de leurs interfaces en entrée. Les informations de la communication de Host A vers ISP2 sont collectées deux fois, la première fois quand il rentre dans le routeur 1 par l'interface Gi1/1 et la seconde fois lorsqu'il entre dans le routeur 2 par l'interface Gi1/3. Il en est de même pour les communications de ISP 2 vers Host A. Au niveau hardware, il n'est pas possible dans ce cas-ci d'éviter la duplication des NetFlow. Elle doit être détectée par l'application qui analysera les enregistrements NetFlow.

Une solution est une application qui collecte les informations et rejette les données dupliquées. Les flux ayant les mêmes adresses IP source/destination, ports source/destination et le même protocole peuvent être considérés comme identiques.

Une autre solution est une application qui collecte les informations nécessaires au filtrage des données collectées (l'adresse IP de l'Exporter et les interfaces d'entrées et de sorties pour chaque flux). Ces données peuvent être utilisées pour filtrer. Dans la figure ci-dessus nous pouvons par exemple exclure les informations du trafic entre le routeur 1 et 2 de l'analyse, l'information étant dupliquée à cet endroit. Le problème est donc résolu en excluant tous les enregistrements NetFlow générés par le router 1 avec l'interface Gi1/1 en entrée et l'interface Gi1/3 en entrée du routeur 2 de l'analyse.

2.6.1.6 Exportation des enregistrements NetFlow à partir d'un périphérique de couche 2

Quand les composants réseaux ne supportent pas le protocole NetFlow, une solution existe. Un programme doit être installé sur un serveur et une sonde NetFlow analyse et génère les enregistrements NetFlow. Le composant doit être configuré et le trafic passant par ses interfaces doit être retransmis vers le serveur.

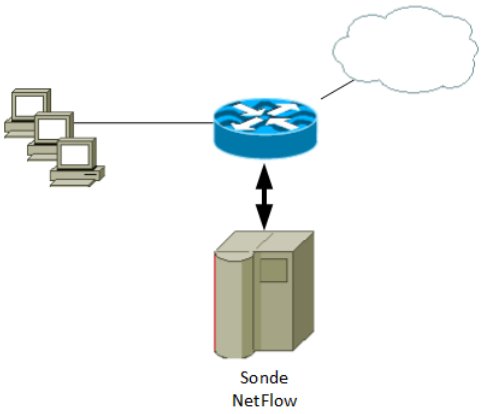


FIGURE 12 – Retransmission du trafic vers la sonde NetFlow

La figure 13 donne une vue plus détaillée de la retransmission au niveau des ports. Le trafic passant par le port Gi0/0 (uplink) est dupliqué ("mirroré") sur le port Gi0/1, ce qui permet de collecter les informations du trafic entrant par les interfaces Fa0/X du périphérique de couche 2 et sortant par l'interface Gi0/0. Il n'est cependant pas possible d'obtenir les informations du trafic local entre les interfaces du composant de couche 2.

Il est indispensable de disposer des deux interfaces réseaux sur le serveur car le port du composant réseau par lequel le traffic est retransmis est inutilisable pour une communication normale et donc il sera impossible d'exporter les données NetFlow. Une seconde interface connectée au composant réseau permettra l'exportation des enregistrements NetFlow. La puissance nécessaire dépend de la grandeur du réseau à analyser. Le type de programme permettant d'effectuer cela est appelé sonde NetFlow (NetFlow probe).

Cette solution comporte des désavantages, un certain nombre d'interfaces supplémentaires sont utilisées et des informations standards du protocole NetFlow sont perdues. Ces informations sont l'adresse IP de l'Exporter, l'interface d'entrée et de sortie ne peuvent pas être utilisées car elles sont indisponibles avec cette méthode. Les applications qui analysent les enregistrements NetFlow sur base de ces informations ne peuvent donc pas être utilisées.

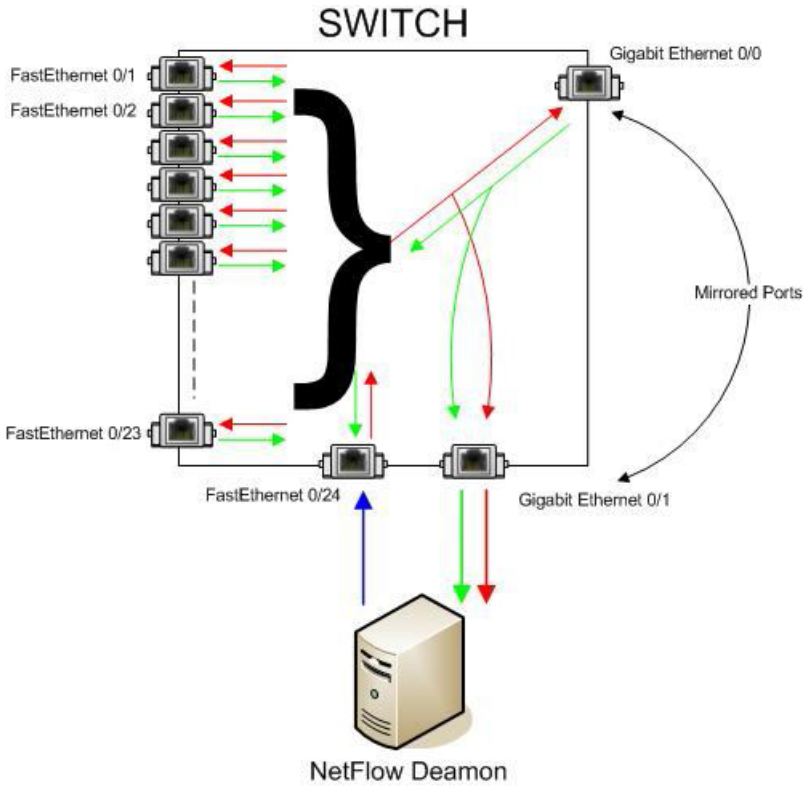


FIGURE 13 – Vue détaillée au niveau des ports (I. Ivanovic [2011])

2.6.2 Configurer les intervalles d'exportation

Les applications traitent les informations de temps (quand a commencé un flux et combien de temps) de deux manières différentes.

La première méthode utilise l'information de temps contenue dans le champ timestamp du NetFlow brut. Ce champ donne comme information le début et la durée du flux.

La deuxième méthode concerne les applications conçues pour traiter une grande quantité d'informations. Elles ne peuvent pas lire le timestamp de chaque flux, ce qui ralentirait fortement le traitement et la taille de la base de données des données traitées. Ces applications définissent et utilisent un intervalle (souvent 5 minutes) et stockent les données agrégées de chaque host du dernier intervalle comme un seul échantillon de 5 minutes.

Dans la figure 14 ci-dessous, nous voyons plusieurs hosts qui génèrent un ou plusieurs flux pendant un intervalle de temps (ici 5 minutes). Au lieu d'enregistrer les données pour chaque flux émis par les différents hosts dans la base de données, seul le total des flux émis par chaque host durant l'intervalle est enregistré.

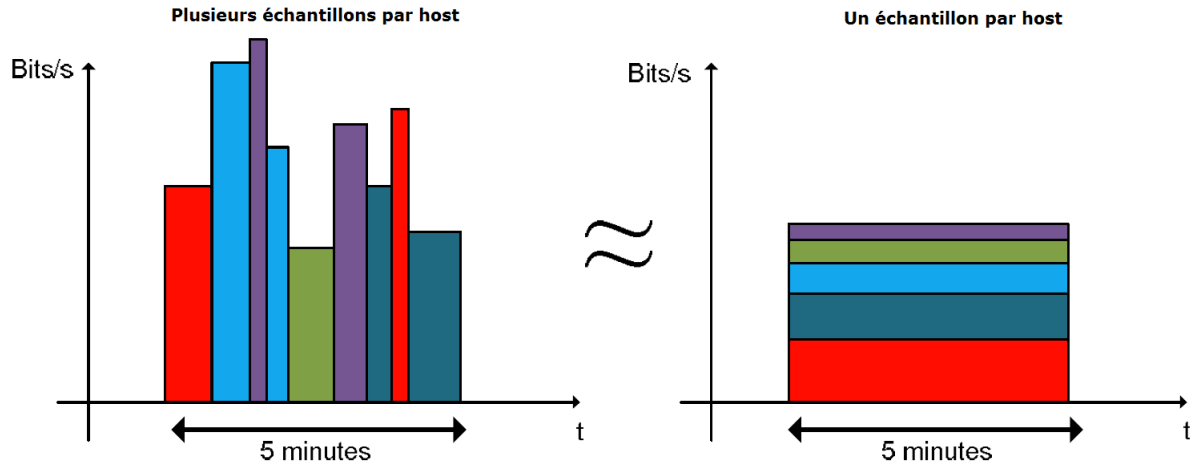


FIGURE 14 – Exemple de stockage des flux dans la base de données (I. Ivanovic [2011] [24])

Lorsqu'on utilise une application qui exploite les données de cette façon, il faut porter une attention particulière à l'intervalle choisi pour l'exportation. L'intervalle de temps choisi pour l'exportation doit être plus court ou égal à l'intervalle utilisé par l'application.

Pour que les données soient correctes en ce qui concerne les flux de longues durées ou pour les flux de très courtes durées, l'intervalle de temps doit être ajusté lorsque la duplication du trafic est configurée.

Dans la figure 15 nous voyons un flux qui dure 20 minutes, les NetFlow le concernant ne sont envoyés qu'à la fin de celui-ci. L'application de l'exemple n'utilise pas le champ avec le timestamp. L'application va considérer que le flux n'a duré que 5 minutes au lieu du temps réel de 20 minutes. Le rapport quantité de données/durée ne correspond pas à la réalité.

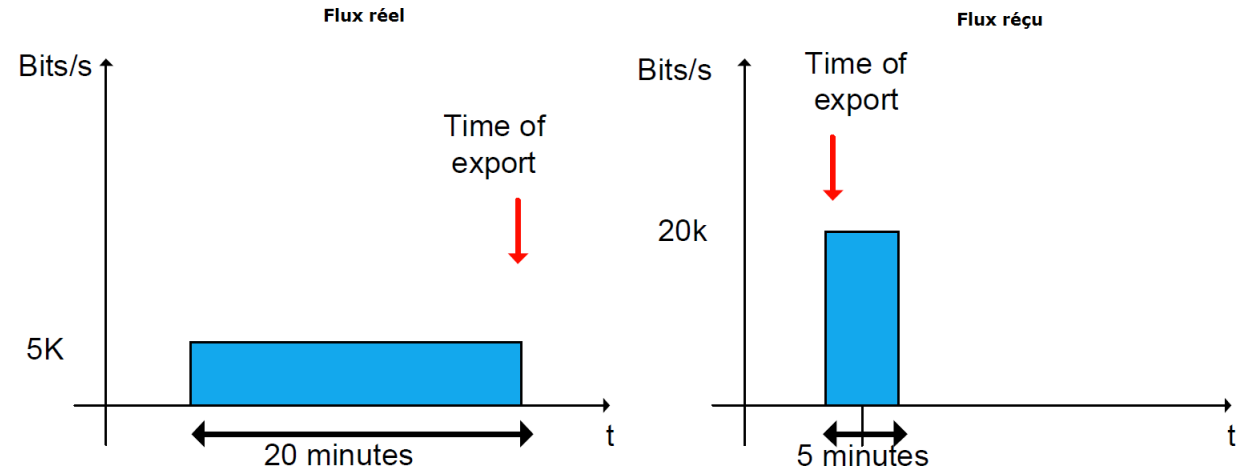


FIGURE 15 – Exemple de NetFlow collectés incorrectement (I. Ivanovic [2011] [24])

Les routers ou switch Cisco permettent de définir des timers pour déterminer qu'un flux est terminé (flow-aging).

Il y en a 3 types :

- Normal aging par défaut 5 minutes ;
- Fast aging réglé sur un temps plus court que le normal aging et en combinaison avec la quantité de paquets transmis. Il permet d'avoir de courtes requêtes (requêtes DND, Ping), les marquer comme complètes, expirées et les exporter rapidement. De cette façon les attaques peuvent être détectées plus rapidement.
- Long aging est utilisé pour marquer les flux de longues durées comme expirés et les exporter vers le collecteur.

Ces timers doivent être configurés correctement pour éviter que le cache ne soit surchargé. En cas d'attaque de déni de services, le cache peut être rapidement rempli. Quand le cache est rempli, toutes les informations contenues dans celui-ci sont exportées (que les flux soient complets ou pas) pour libérer le cache. Ce processus peut demander une charge CPU conséquente.

Le fast aging des NetFlow permet de marquer les requêtes de courte durée comme expirées et de les exporter, ce qui libère le cache.

2.7 Détection d'activités anormales avec NetFlow

Nous allons voir comment on peut détecter des activités réseaux "anormales" à partir des NetFlow à l'aide de plusieurs techniques. Afin de présenter les différentes techniques, nous nous sommes basés sur les articles de Yiming Gong [2010] [21] et de Bingdong Li, Jeff Springer, George Bebis, Mehmet Hadi Gunes [2013] [26]. Les enregistrements NetFlow permettent deux types de détection. Le premier type de détection est basé sur le comportement. On compare le trafic réseau à un modèle représentant le trafic considéré comme normal. Le deuxième type de détection est basé sur la signature d'attaques. On compare le trafic à ces signatures.

2.7.1 Référence et top N

Un modèle doit être créé afin de décrire ce qu'est une activité réseau "normale", il est basé sur l'historique des activités du réseau. Tout le trafic qui se retrouve en dehors de ce modèle sera considéré comme anormal. Au début, lorsqu'un modèle est défini, nous ne pouvons pas garantir que du trafic "anormal" ne s'y trouve pas puisqu'il est basé sur l'historique du trafic réseau. Des anomalies peuvent donc passer inaperçues. Le modèle peut aussi considérer des activités comme anormales si celles-ci n'étaient pas présentes auparavant alors qu'elles sont parfaitement licites.

On peut donc s'attendre à devoir apporter beaucoup de modifications au début et de moins en moins avec le temps. Le modèle sert de référence.

2.7.1.1 Top N sessions

Si un host produit un grand volume de demandes de connexions vers une destination ou un ensemble de destinations par rapport au modèle de référence, il sera considéré comme un trafic réseau anormal. La raison la plus classique est la présence d'un vers, d'une attaque DoS/DDoS, d'un scan réseau ou d'abus.

Dans une situation normale, lorsqu'un host se connecte à Internet, la fréquence des requêtes doit rester relativement normale. Par contre, si le client est infecté par un vers, on peut souvent observer un grand nombre de demandes de connexions vers l'extérieur afin de tenter d'infecter d'autres clients.

Si un script ayant pour but de scanner un bloc d'adresses afin de détecter des vulnérabilités est lancé, nous verrons un grand nombre de requêtes vers une et même IP.

La méthode utilisant le Top N session peut aussi être utilisée pour détecter toutes sortes d'abus réseau. Par exemple, dans un temps donné, chaque poste effectuant un nombre de requêtes significativement plus important que la normale peut être considéré comme un spammer ou une machine étant infectée par un vers.

2.7.1.2 Top N données

On peut utiliser la même technique avec les données. Un classement des hosts transférant le plus de données vers ou en dehors de l'entreprise durant un laps de temps défini doit être établi. Ce classement doit être relativement stable. Si un hôte apparait soudainement dans le classement, une alerte doit être émise.

Cette augmentation soudaine de transfert de données pourrait être liée au fait que le poste est contaminé par un ver.

2.7.2 Modèles (Pattern matching)

La méthode basée sur les modèles est une autre technique permettant d'identifier des activités réseaux "anormales" et est elle aussi basée sur l'analyse des flux. Un modèle est établi en prenant certains champs du flux (adresse IP, port,...) et en leur donnant des valeurs spécifiques. Ces valeurs formeront les critères du modèle. Les flux correspondant à ce modèle seront donc considérés comme suspects.

Les champs les plus couramment utilisés pour établir le modèle sont les adresses IP source et destination et les numéros de port source et destination mais tous les champs peuvent être utilisés.

2.7.2.1 Port (Port matching)

En général, chaque attaque vise un port spécifique, on peut donc trouver tous les flux où le port de destination est égal au port spécifique afin d'identifier une possible attaque. Cette méthode est très simple à implémenter et peut être utilisée dans de nombreux cas.

2.7.2.2 Adresse IP (IP adress matching)

Nous pouvons spécifier un range d'adresses IP en fonction de certains critères comme, par exemple, des adresses IP non routées réservées par l'IANA (Organisation qui gère l'espace d'adressage IP d'internet), elles ne sont donc pas censées être présentes dans les flux. Dans ce cas la, nous ne pourrons donc pas retracer l'origine à l'aide l'adresse IP car il s'agit d'une adresse usurpée. Il faudra utiliser un autre champ du flux afin de l'identifier.

Ces deux techniques nous permettent de construire des modèles sur base d'attaques (connues) où l'on connaît le port utilisé ou le serveur d'un Botnet par exemple.

3 Pratique

Nous allons dans cette partie évaluer plusieurs logiciels qui offrent une interface graphique permettant l'analyse des NetFlow.

Afin d'effectuer cette comparaison, nous allons nous inspirer d'une méthodologie développée par USmax[18] permettant l'évaluation de produits "Commercial off-the-shelf" (COTS - composant fabriqué en grande série et non pour un projet en particulier [Def. Wikipédia] [3]). Cette méthodologie utilise une approche basée sur les buts et les objectifs des clients. L'avantage d'utiliser cette méthode est qu'elle peut s'adapter à divers contextes en redéfinissant les objectifs et les buts qui lui sont propres. Nous appliquerons donc cette méthode à notre contexte après l'avoir définie.

3.1 La méthodologie d'évaluation

Comme indiqué dans l'introduction de ce chapitre, notre méthodologie s'inspire de celle développée par USmax. Les différentes phases sont les suivantes :

- Phase I : **Les exigences :** nous identifierons les personnes concernées par l'utilisation du produit (stakeholders) et décrirons leurs exigences, ainsi que celles propres au contexte ;

- Phase II : **Les critères d'évaluation :** nous citerons ici les différents critères d'évaluation qui permettront d'élire les 3 produits qui correspondront le mieux aux exigences retenues. Ils seront basés sur les exigences ;

- Phase III : **Evaluation des produits :** nous appliquerons ici les critères d'évaluation qui permettront d'attribuer un certain nombre de points aux produits ;

- Phase IV : **Evaluation finale :** les 3 produits seront comparés au niveau de leur performance et de leurs fonctionnalités, afin de fournir la meilleure solution. Un modèle générique sera développé, afin de le comparer aux différents produits.

Nous allons maintenant appliquer cette méthodologie à notre contexte (UCL).

3.1.1 Phase I : Les exigences

Les stakeholders sont le responsable sécurité du système d'information (RSSI) et l'auteur de ce mémoire.

Le produit retenu devra permettre d'analyser les NetFlow aussi bien en temps réel qu'en mode offline et posséder une interface graphique. On doit pouvoir analyser des données récoltées dans le passé. L'objectif de cette analyse sera de pouvoir repérer des anomalies dans le trafic réseau.

Les entreprises exploitant le produit devront être réputées, ce qui nous donne certaines garanties par rapport à la qualité du produit et nous évite de tester des produits non aboutis.

Afin de faciliter la mise en place du produit et la résolution d'éventuels problèmes, une communauté active ou un support facilement accessible sont indispensables.

Notre contexte nous impose aussi certaines exigences. Le serveur sur lequel sont récoltés les NetFlow est sous Red Hat. Les NetFlow étaient jusqu'au mois de mai récoltés en version 7 et maintenant le sont en version 9. Le réseau actuellement en IPv4 passera à court terme en IPv6. Une compatibilité avec le format NetFlow version 9 et l'IPv6 est donc indispensable.

3.1.2 Phase II : Les critères d'évaluation

Les critères basés sur les exigences des stakeholders sont les suivants :

- analyse offline (possibilité d'analyser des NetFlow déjà collectés) ;
- analyse temps réel ;
- permettre la détection d'anomalies ;
- utilisés par des entreprises réputées ;
- communauté active ou support facilement accessible.

Les critères basés sur les exigences liées au contexte sont les suivants :

- compatible Red Hat ;
- compatibilité IPv6 ;
- compatibilité format NetFlow 5,7,9.

3.1.3 Phase III : Evaluation des produits

Chaque critère sera évalué et les 3 produits ayant obtenu le plus haut score seront retenus. Il nous restera donc les 3 produits les plus proches des exigences.

Système de cotation :

- 0 - incompatible ou pas disponible
- 1 - compatible ou disponible mais adaptation nécessaire
- 2 - disponible ou compatible conformément aux exigences
- 3 - dépasse les exigences

	NFSEN	CALLI-GARE	NTOPNG	MANAGE ENGINE	PLIXER	SOLAR WINDS
Analyse offline	3	2	2	2	2	2
Analyse temps réel	2	2	2	2	2	2
Détection d'anomalies	2	3	2	3	2	2
Partenaires	2	2	2	2	2	2
Support	3	0	3	2	2	2
Compatibilité Red Hat	2	2	2	2	0	0
Compatibilité NetFlow v5,7,9	2	2	2	2	2	2
Compatibilité IPv6	2	2	2	2	2	2
Gratuité	2	0	2	0	0	0
Total	20	15	19	17	14	14

TABLE 2 – Comparaison produits

Les 3 produits retenus sont :

- NfDump / NfSen ;
- Ntopng ;
- NetFlow Analyzer de ManageEngine ;

3.1.3.1 NFDump / NFSen

NFDump et NFSen sont sous licence libre. NFDump/ NFSen est la solution préconisée par le CNRS (Centre National de la Recherche Scientifique) pour la mise en place d'un collecteur NetFlow.

NfDump est un ensemble d'outils permettant de collecter et d'analyser les NetFlow. Il supporte les versions 5,7 et 9 de NetFlow.

Les deux outils principaux sont nfcapd et nfdump.

Le premier est un process permettant de lire les NetFlow venant du routeur via le réseau et de les stocker dans des fichiers sur le serveur. Les fichiers sont créés à intervalles réguliers (par défaut 5 minutes). Nfcapd peut lire les versions 5, 7 et 9 de NetFlow de façon transparente et est compatible IPv6.

Le second lit les fichiers générés et stockés par nfcapd. Il permet de les analyser afin de dégager des informations ou des statistiques et d'afficher les résultats.

NfDump permet d'analyser les données NetFlow aussi bien dans le passé qu'en "live". La seule limite de l'historique est la capacité du disque dur.

Les données sont stockées sur le disque avant d'être analysées. Comme mentionné plus haut des fichiers sont générés à intervalles réguliers. Une fois le temps écoulé, le fichier est renommé à l'aide de son time stamp (nfcapd.YYYYMMddhhmm).

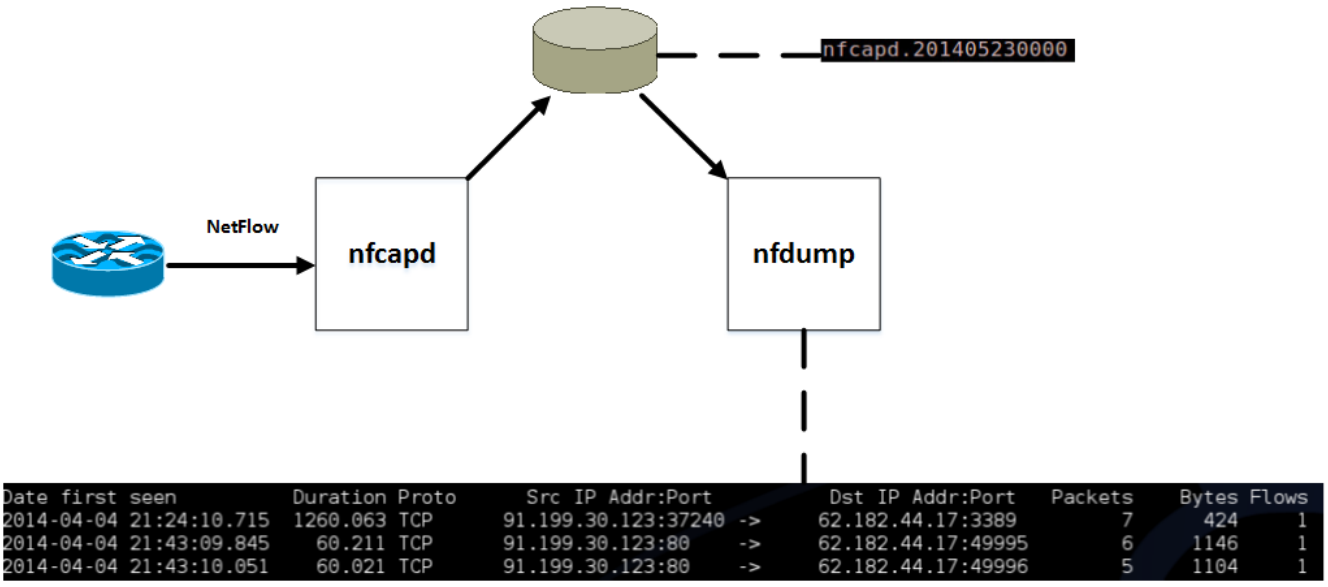


FIGURE 16 – NfDump

NFSen est une interface graphique web pour l'outil NFDump. Il permet donc de représenter les données NetFlow graphiquement en utilisant RRD (Round Robin Database) et de les parcourir facilement.

Les NetFlow peuvent être analysés dans un intervalle de temps spécifique.

Il permet également de créer différents types de profils. Nous pouvons créer des profils "live" et les différencier en fonction de certains critères comme, par exemple, le protocole utilisé. Des profils d'historiques peuvent aussi être créés.

Une autre fonctionnalité offerte est la mise en place d'alertes basées sur plusieurs conditions comme le dépassement de certains seuils (volume ou nombre de connexions dans un intervalle de temps).

NfSen donne aussi la possibilité de développer des plugins pour traiter les NetFlow à intervalles réguliers.

[Source : sourceforge.net] [10], [9]

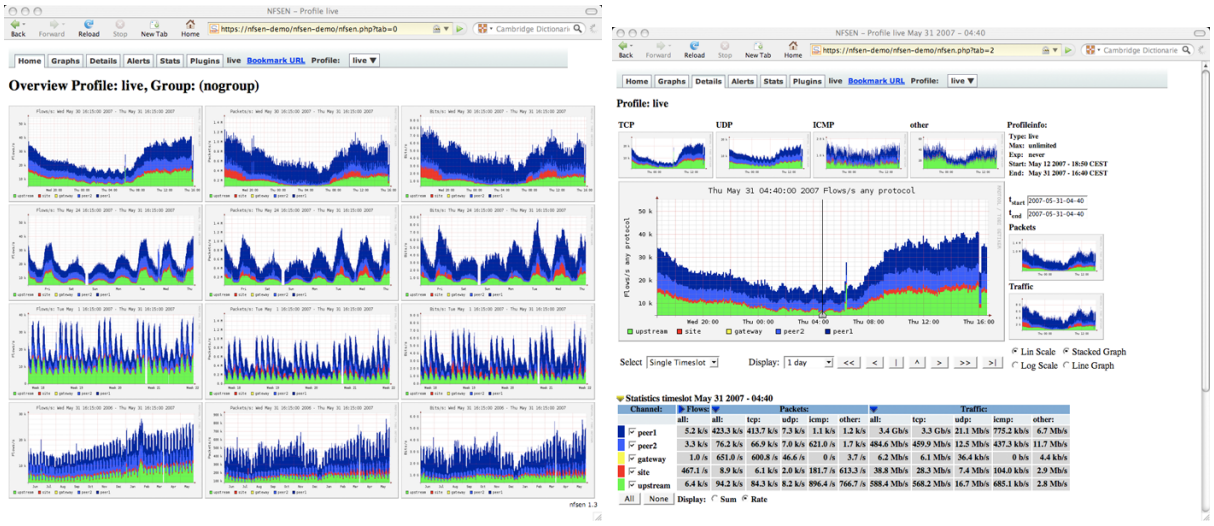


FIGURE 17 – Illustration des différentes vues de l'interface NfSen (chaque couleur représente une source NetFlow différente).

3.1.3.2 Ntopng

Ntopng utilise Nprobe pour lire les NetFlow et les stocker sous forme de fichiers sur le disque dur ou dans une base de données MySQL ou SQLite. Le fonctionnement est similaire à celui de nfcapd, vu précédemment. Nprobe peut collecter les versions de NetFlow 5, 7, 9 ou IPFIX et est compatible IPv6.

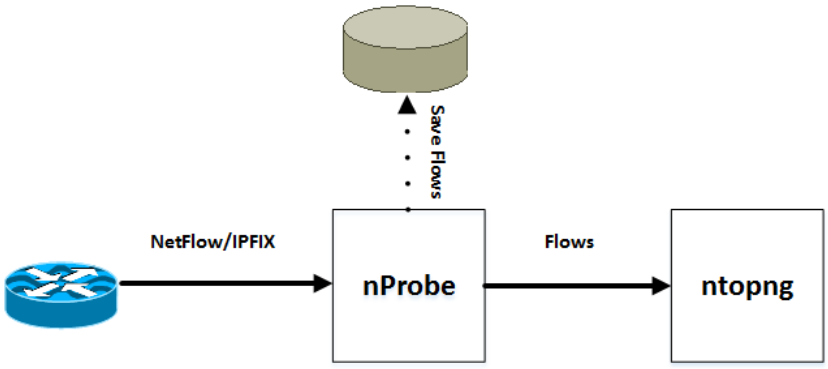


FIGURE 18 – nProbe / ntopng (mode sonde)

Nprobe peut être utilisé en 3 modes différents.

Le premier comme une sonde NetFlow, il transmet les flux NetFlow au collecteur (ex : Ntopng) et peut les stocker.

Le deuxième comme un collecteur, il récupère les flux et les stocke.

Le troisième comme un proxy, il récupère les flux, les stocke ou non, les convertit dans le format souhaité et les envoie au collecteur. Il peut convertir de/en IPFIX/NetFlow 5, 7, 9. Ce mode permet des upgrades en douceur vers des versions plus récentes de NetFlow. Des informations seront parfois perdues lors de la conversion en fonction des versions utilisées.

Ntopng est une version de nouvelle génération de ntop, d'où l'ajout de "ng". Ntop est un outil d'analyse en ligne de commande pour Unix. Ntopng est basé sur lipcap qui est une librairie portable C/C++ pour la capture du trafic réseau. Ntopng a été écrit pour pouvoir s'exécuter sur les toutes les plateformes Unix, MacOSX et Win32.

Une interface graphique accessible via un browser Web permet de naviguer à travers les informations du traffic réseau fournies par Ntopng et affiche le statut du réseau. L'interface graphique facilite fortement la configuration et l'administration.

Ntopng permet de trier le trafic en fonction des protocoles utilisés, de montrer le trafic réseau et les hosts IPv4/v6 actifs. Il peut grâce au framework nDPI mettre en évidence les applications utilisées sur le réseau à partir du protocole utilisé par celles-ci. Ntopng permet aussi de montrer la distribution du trafic IP entre les différents protocoles, d'analyser et classer le trafic en fonction de la source et la destination et de générer une matrice mettant en évidence "qui parle avec qui".

[Source : ntop.org] [7]

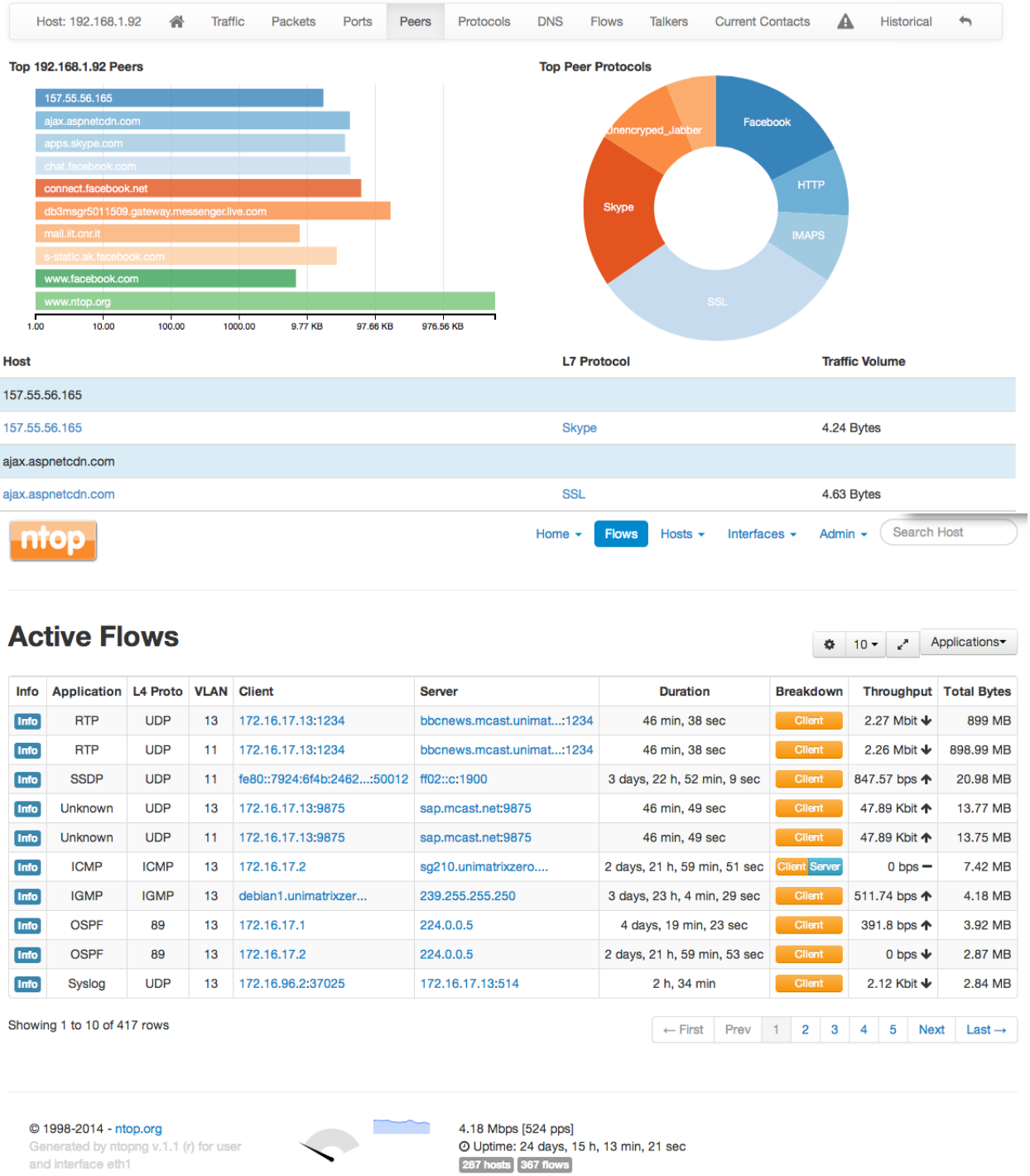


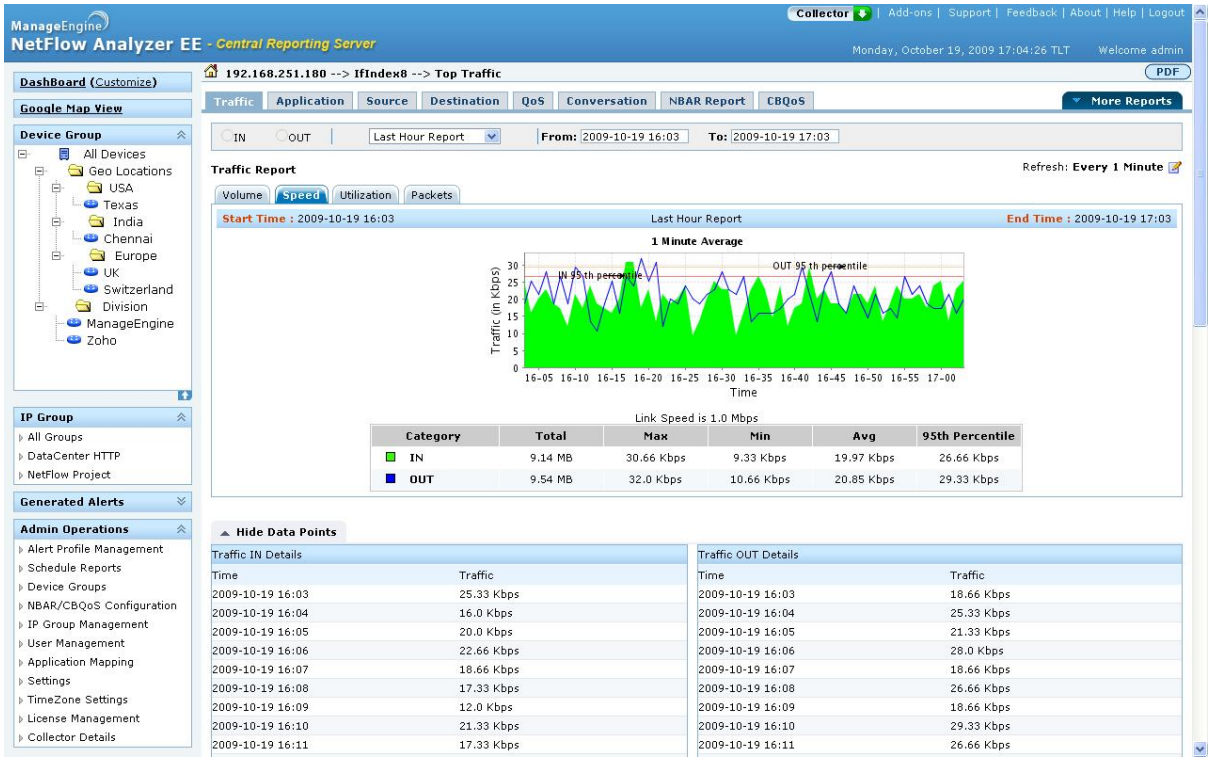
FIGURE 19 – Illustration de l'interface ntopng (ventilation des destinations d'un poste)

3.1.3.3 NetFlow Analyzer de ManageEngine

NetFlow Analyzer est une solution payante contrairement aux deux solutions précédentes.

NetFlow Analyzer est une solution unifiée qui collecte, analyse et génère des rapports sur l'utilisation du réseau par les utilisateurs. Il permet d'avoir une vision globale de la bande passante et des tendances d'utilisation de celle-ci. Comme les deux solutions précédentes, l'interface graphique est une interface Web.

Il est compatible avec les NetFlow version 5, 7 , 9 et IPFIX et IPv6.



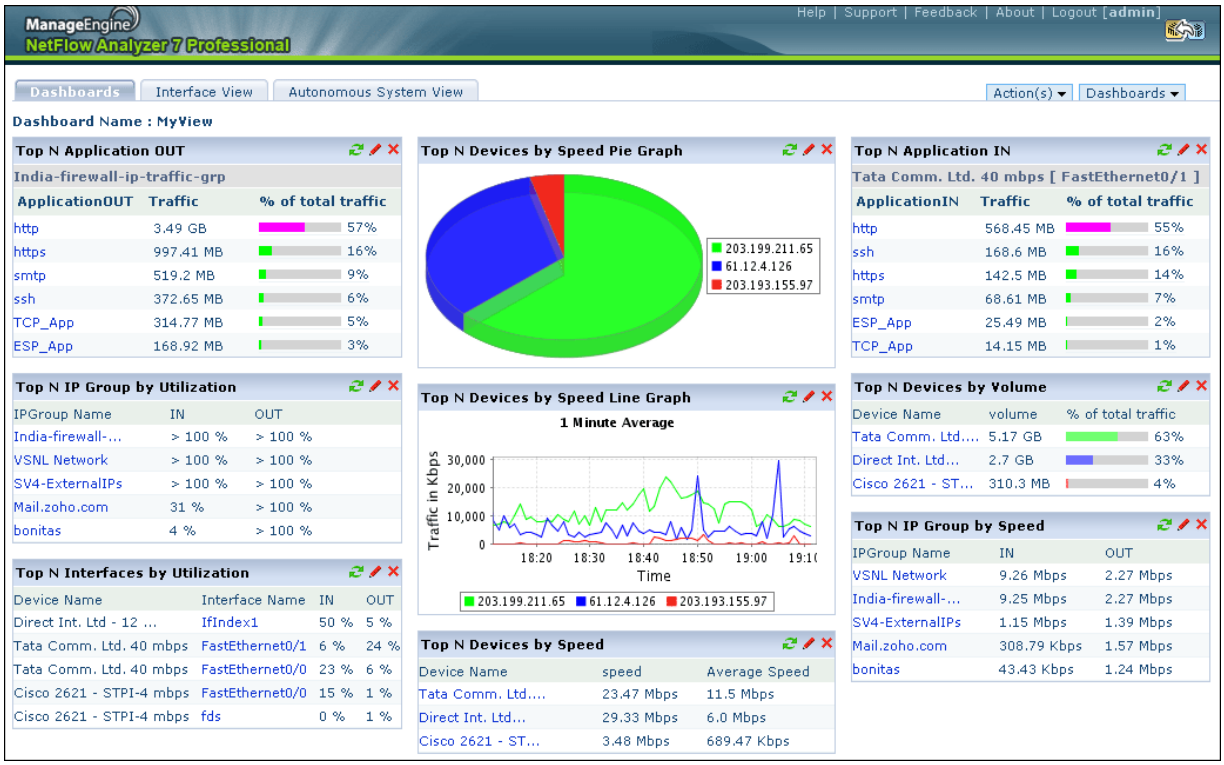


FIGURE 20 – Illustration de l'interface NetFlow Analyser (statistiques Top N d'un réseau)

Il permet d'analyser la bande passante réseau et d'en dégager les tendances du trafic. NetFlow Analyser peut aussi détecter un grand nombre de menaces de sécurité et d'anomalies, mettre en évidence les applications qui utilisent la bande passante. Des alertes peuvent aussi être configurées en fonction de certains critères.

NetFlow Analyzer dispose d'un module dédié à la sécurité, ce qui n'est pas le cas des deux autres produits.

[Source : manageengine.com] [5]

3.1.4 Premières constatations

Les 3 solutions utilisent une interface web, ce qui facilite une utilisation et une administration à distance.

ntopng et NetFlow Analyser offrent un interface plus moderne et des fonctionnalités plus avancées nativement. On peut par exemple très facilement observer l'utilisation de la bande passante par les différentes applications présentes sur le réseau pour autant qu'elles soient prises en charge par l'application.

NfSen offre nativement moins de fonctionnalités mais paraît plus facilement personnalisable que les deux autres produits. Ntopng et NetFlow Analyser semblent être des produits clés en main contrairement à NfSen où la création manuelle de filtres est nécessaire afin de dégager des informations. Nous sommes donc amenés à trouver le bon compromis entre la facilité d'utilisation et la personnalisation possible.

3.1.5 Phase IV : Evaluation finale

Nous allons ici comparer les 3 produits au niveau de leur performance et de leurs fonctionnalités. Le but sera de garder le produit permettant de répondre le mieux aux exigences techniques.

3.1.5.1 Données de test

Afin de comparer les différentes solutions entre elles, nous devons nous procurer un jeu de données NetFlow qui sera strictement identique entre les différents tests. Cet aspect n'a pas été facile et a pris beaucoup de temps. Des problèmes ont été rencontrés à plusieurs niveaux. Nous devions trouver des données pouvant être rejouées à l'infini et ayant les caractéristiques des flux "live" (débit, données contenues). Pour ce faire plusieurs solutions ont été imaginées.

A. Redirection de port

La première solution est d'effectuer une redirection de port sur le routeur, afin que les NetFlow soient envoyés sur plusieurs ports différents de la machine les collectant.

Cette solution semblait idéale car elle permet d'utiliser le même flux NetFlow pour les tests des trois logiciels et elle ne demande que très peu de configuration (seulement l'ajout de quelques lignes dans la configuration du routeur). Cependant lors des tests, la machine devenait quasi inutilisable. En effet, elle ne dispose que d'une seule interface réseau par laquelle transitait le flux NetFlow triplé ainsi que l'accès à distance SSH. Cette interface était saturée. La solution aurait été d'ajouter une interface réseau mais n'ayant pas d'accès physique à la machine, cela n'a pas été possible. Cette solution a eu un autre impact négatif au niveau du router. La charge CPU de ce dernier s'est vue fortement

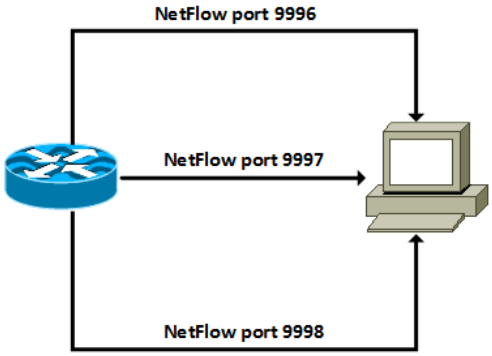


FIGURE 21 – Redirection de port

augmenter. La solution ne permettait pas non plus de rejouer le même jeu de données plusieurs fois. C'est pour ces raisons que cette solution n'a pas été retenue.

B. Duplication des NetFlow

Afin de résoudre le problème de la saturation de l'interface réseau rencontré dans la solution précédente, nous avons imaginé une autre solution dont le concept est fort semblable au premier. Au lieu d'envoyer les NetFlow sur trois ports différents à partir du router, les NetFlow sont envoyés une seule fois sur la machine et sont dupliqués par celle-ci.

Un programme (eg : Samplicator) permet d'effectuer l'opération d'une manière assez simple. Samplicator écoute sur un port donné et renvoie une copie du flux reçu à plusieurs destinations. Une option permet d'effectuer du spoofing d'adresse, ce qui permet de le faire passer pour la source et non un relais.

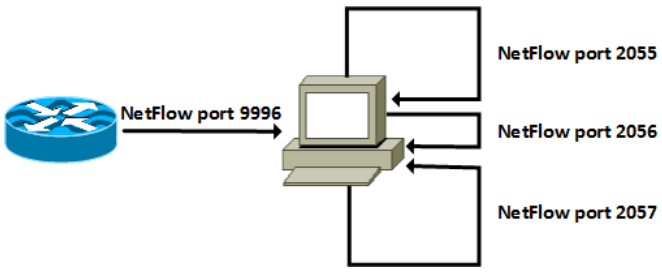


FIGURE 22 – Duplication des NetFlow

Le problème avec cette solution est qu'elle demande beaucoup de ressources. Une partie des resources est utilisée par Samplicator et une grosse partie par les trois analyseurs NetFlow qui tournent en parallèle. Même si ce problème n'est pas bloquant, il dégrade fortement les performances du serveur. De plus, on aimerait pouvoir rejouer plusieurs fois les mêmes données, pour effectuer nos différents tests. Les deux premières solutions ne le permettent pas.

C. Rejouer les NetFlow

Nous voulons pouvoir rejouer les mêmes NetFlow plusieurs fois. La solution est donc de rejouer des NetFlow précédemment stockés sur le serveur. Comme nous l'avons indiqué plus haut, une routine stockant une copie des NetFlow sur le serveur est déjà mise en place. Nous pouvons donc les utiliser et les rejouer avec la commande "flow-send" de la librairie flow-tools à l'aide d'un script.

```
#! /bin/bash

for flow in $(find -L /disks/data/nfr/ -type f -newer /home/cluycx/netflow-scripts/t1
-and -not -newer /home/cluycx/netflow-scripts/t2 -print | sort -n); do
echo $flow
gpg -d $flow | xz -d |flow-send -x 500 0/localhost/9997

done

exit
```

Code 3 – nfreplay.sh

Le script permet de rejouer les NetFlow d'un intervalle de temps spécifique. L'intervalle est défini à l'aide de deux fichiers t1 et t2 dont le timestamp a été modifié. L'intervalle est compris entre le timestamp de t1 et celui de t2.

Une question s'est posée au moment où la solution a été imaginée. Les NetFlow rejoués contiennent-ils toujours exactement la même information que lorsqu'ils sont capturés en live ? Pour répondre à cette question, nous avons effectué une série de tests (top 10 volumes échangés, top 10 des ports utilisés, top 10 des destinations, top 10 des ports utilisés, ...), sur des NetFlow couvrant une période de 24 heures. Les résultats retournés de ces tests sont identiques entre les NetFlow rejoués et ceux collectés en live. Aucune information pertinente pour nos besoins n'est perdue.

La seule différence provient du graphique généré. La quantité d'informations est la même tant "en live" que "rejoué". Cependant, dans le premier cas, les données sont réparties sur une durée de 24 heures et dans le second cas, sur une durée d'environ 4 heures. Le résultat génère un graphique moins précis. Nous pourrions corriger cela en augmentant le temps d'envoi lorsque les NetFlow sont rejoués. Les graphiques n'étant pas faussés et les données récoltées étant identiques, cette différence ne posera pas de difficultés pour les tests qui nous intéressent ici. En effet, nous vérifierons que les différents logiciels génèrent les mêmes résultats au niveau des données. Les NetFlow seront capturés "en live" par le produit qui sera mis en production. Nous ne perdrons pas de temps à corriger cela.

Bien que dans un premier temps nous avions décidé de ne pas corriger l'aspect temps lorsque nous rejouons les NetFlow, une solution a quand même été mise en place par la suite. La solution imaginée a été de limiter le débit lors du replay à l'aide de la commande "pv". Celle-ci permet de limiter le débit au travers d'un pipe.

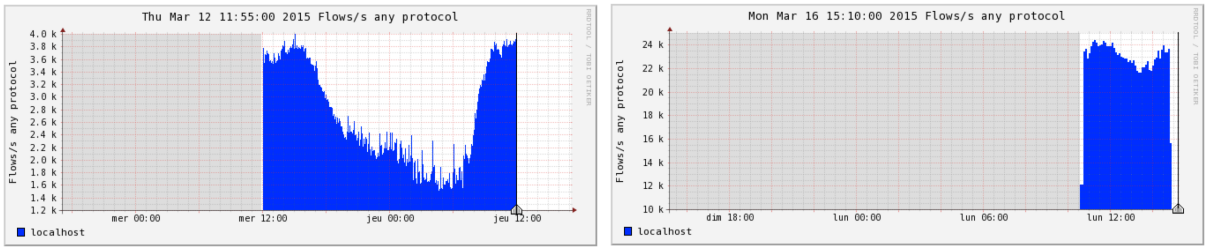


FIGURE 23 – Live vs Replay

3.1.5.2 Evaluation

Nos premiers tests ont eu pour but de vérifier que les trois produits retenus nous donnent les mêmes résultats. Nos tests ont été effectués sur un échantillon d'une durée de 24h (rejoué).

Les différents tests :

- Top 10 hosts (volume) ;
- Top 10 ports ;
- Top 10 IP destinations ;
- Top 10 IP sources ;
- Top 10 protocoles ;
- Recherche d'adresses IP

Les trois produits nous ont donné des résultats similaires aux différents tests. Ces résultats nous permettent d'affirmer que nous pourrons détecter les anomalies choisies dans le réseau à l'aide des trois produits.

En effet, si on se base sur les différentes techniques de détection vues à la section 2.7, on pourra dans un premier temps déterminer un modèle de référence et ensuite comparer ce modèle afin de détecter des écarts qui pourraient être des menaces.

Comme nous l'avons indiqué plus haut, la qualité des alertes dépendra fortement de notre base de référence. Si notre base de référence contient déjà des menaces, nous aurons à faire à des faux négatifs qui seront quasi impossibles à détecter. En ce qui concerne les faux positifs nous pourrons les diminuer en affinant notre modèle de référence au fur et à mesure.

Nous avons constaté qu'au niveau de la fiabilité de la détection, les 3 produits sont au même niveau. Nous allons maintenant regarder au niveau des fonctionnalités et de la performance, afin de les départager.

En nous inspirant de la méthode de l'article de Emmanuel S. Pilli, R.C. Joshi, Rajdeep Niyogi [2010] [28], nous allons développer un modèle générique du produit. Nous comparerons ensuite ce modèle aux différents produits, afin de mettre en avant les lacunes des produits face à celui-ci.

NFSen nous permet de définir des filtres sous forme de plugins qui peuvent générer des alertes en fonctions de critères. La combinaison de ces critères est quasi infinie. Nous pouvons par exemple créer des filtres générant une alerte lorsqu'un nouvel hôte apparaît dans le top N du volume par host, lorsqu'un host communique avec une adresse IP spécifique qu'on considère comme illicite, lorsqu'un nombre de connexions dans un laps de temps est dépassé par un host, etc.. Le désavantage est que la qualité de la détection dépend fortement de la qualité des filtres mis en place. Les filtres doivent être adaptés en permanence et de nouveaux créés pour détecter de nouvelles anomalies.

Ntopng ne permet pas de paramétrer des alertes. Si on veut recevoir des alertes, on peut exporter les résultats au format JSON et les exploiter à l'aide d'un autre logiciel pour générer des alertes. Bien que Ntopng permette d'obtenir les mêmes statistiques que les deux autres produits, il est plus orienté vers le monitoring de l'utilisation de la bande passante. Il permet de définir certains filtres mais de manière assez limitée.

NetFlow Analyzer dispose d'un module appelé Advanced Security Analytic Module (ASAM) permettant de détecter des anomalies réseau, attaque DOS ou encore des scans non autorisés. La détection est basée sur des algorithmes. On peut également configurer des alertes basées sur ces algorithmes ou en fonction de certains critères (IP source/-destination, port source destination, nombre de connexion dans un intervalle de temps, etc..).

Deux produits ressortent donc du lot : NFSen et NetFlow Analyzer. La détection basée sur les filtres nous donnera les mêmes résultats puisque les tests effectués nous ont donné des résultats identiques. La qualité de la détection dépendra donc de la qualité des filtres et du modèle de référence.

L'avantage de NFSen est qu'il est gratuit et qu'il est assez facile de développer des plugins permettant de lever des alertes mettant en pratique les techniques de détection vues. NetFlow Analyzer est payant mais met à disposition un outil consacré à la détection de menaces, tout en nous offrant la possibilité de configurer des filtres générant des alertes.

NetFlow Analyzer intègre un module dédié à la détection d'anomalies, ce qui lui donne un avantage par rapport à NFSen. Après l'avoir testé, nous ne sommes pas convaincus de sa plus value pour nos besoins. ASAM génère beaucoup d'alertes dites "critiques". Nous avons vérifié quelques unes de ces alertes.

Pour effectuer ces vérifications nous avons effectuer un scan des postes concernés par l'anomalie à l'aide de CurrPorts de NirSoft. Ce logiciel permet d'obtenir la liste des connexions TCP/IP, le port utilisé par celles-ci et le process concerné. Dans un cas c'était DropBox, dans un autre Facebook et le dernier des services cloud. Il a aussi considéré les services de certains serveurs comme illicites alors qu'ils sont parfaitement licites. Par contre, l'utilisation d'un peer-to-peer a pu être détectée. Ce module génère donc beaucoup de faux positifs et il est difficile d'en dégager le vrai du faux vu la quantité d'alertes. Ce module ne nous a donc pas donné satisfaction. Nous sommes conscients qu'avec plus de

temps et de réglages, il serait surement possible de dégager d'avantage d'efficacité de ce module.

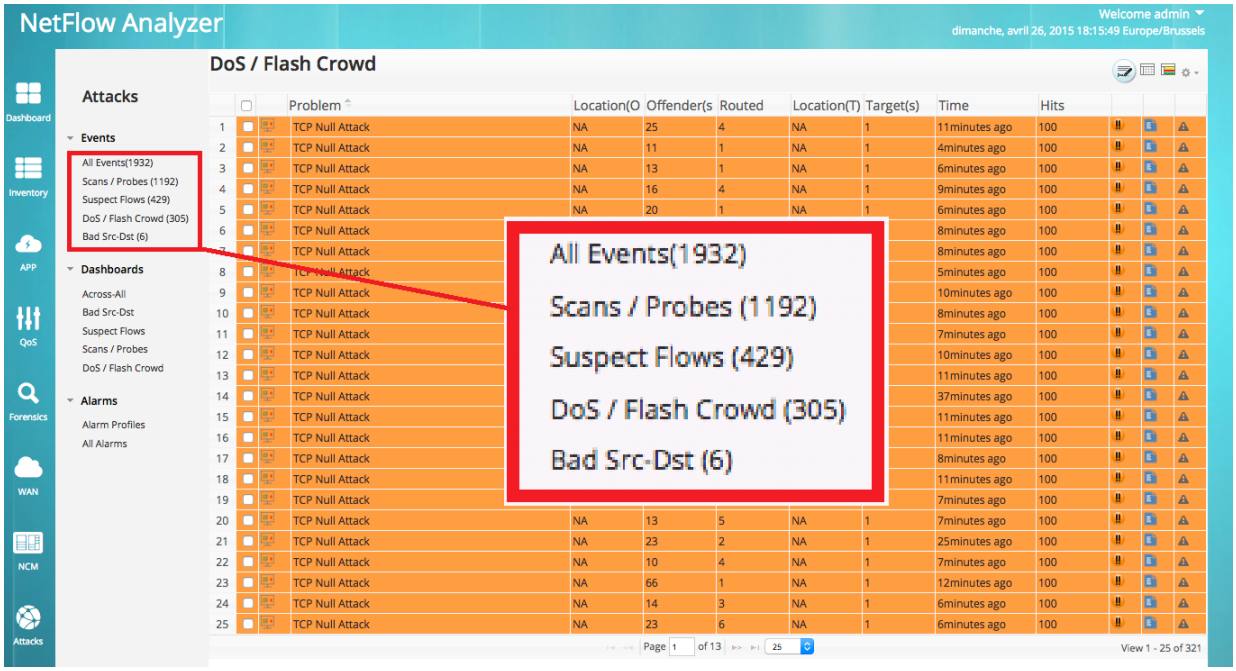
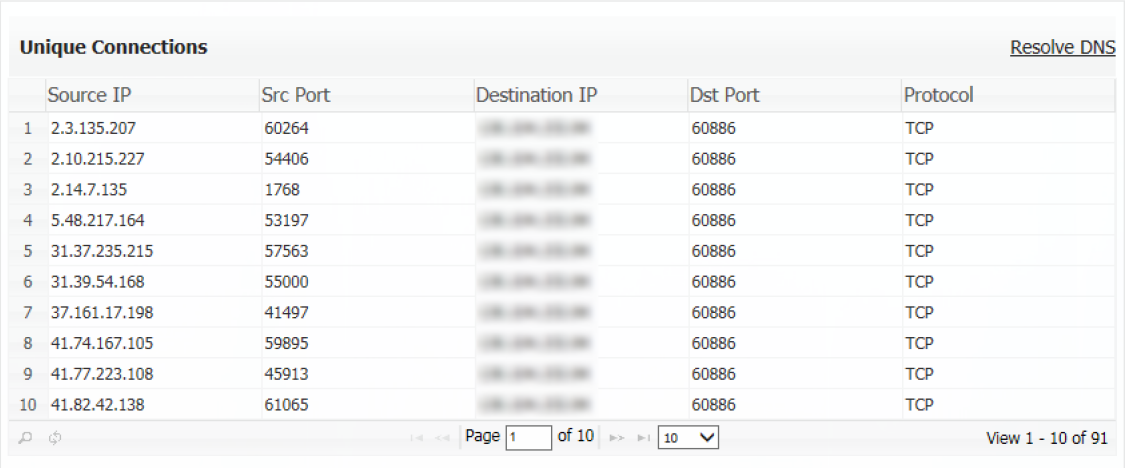


FIGURE 24 – Menaces détectées par ASAM



Unique Connections Resolve DNS

	Source IP	Src Port	Destination IP	Dst Port	Protocol
1	2.3.135.207	60264	[redacted]	60886	TCP
2	2.10.215.227	54406	[redacted]	60886	TCP
3	2.14.7.135	1768	[redacted]	60886	TCP
4	5.48.217.164	53197	[redacted]	60886	TCP
5	31.37.235.215	57563	[redacted]	60886	TCP
6	31.39.54.168	55000	[redacted]	60886	TCP
7	37.161.17.198	41497	[redacted]	60886	TCP
8	41.74.167.105	59895	[redacted]	60886	TCP
9	41.77.223.108	45913	[redacted]	60886	TCP
10	41.82.42.138	61065	[redacted]	60886	TCP

Page 1 of 10 10 View 1 - 10 of 91

FIGURE 25 – Détection P2P

NFSen est donc le produit le plus efficace et le plus adapté aux exigences. NFSen est le plus léger au niveau de l'occupation mémoire et CPU. Il est également le plus modulable.

	NFSEN	**NTOPNG**	**NETFLOW ANALYZER**
Filtres personnalisables	Oui	Oui	Oui
Granularité des filtres	+++	+/-	+/-
Alerting	Oui	Non	Oui
Facilité d'utilisation	+/-	++	++
Ergonomie	-	++	++
Facilité de mise en place	-	+	+++
Stabilité	++	++	+ (après augmentation mémoire JVM)
Personnalisation	+++	-	-
Fiabilité	++	++	++ (ASAM (–) à confirmer)
Consommation ressources	+++	+	+
Fonctionnalités des base	-	+	++
Cohérence aux exigences	++	+/-	+/-

TABLE 3 – Comparaison finale

3.2 Production

3.2.1 Mise en place de NFDump / NFSen

La première étape pour la mise en place est de configurer le routeur pour que les NetFlow soient exportés vers le port 9996 de notre serveur.

1) Installation de NFDump 1.6.12.
2) Installation des pré-requis pour NFSen (PHP5, Perl, librairie rrdtool).
3) Installation de NFSen 1.3.6.
4) Configuration du collecteur via le fichier de configuration nfsen.conf.
5) Configuration d'IPtable (ouverture du port 80 pour l'interface web et du port 9996 pour la réception des NetFlow).
6) Démarrage du Daemon NFSen via la commande "nfsen start".

3.2.2 Exploitation

Nous allons dans cette partie utiliser l'outil NFSen de façon plus avancée. Nous allons mettre en place notre premier filtre, notre première alerte et notre premier plugin. Le but est de montrer les étapes nécessaires à la mise en place et les informations qui peuvent être mises en avant grâce à eux.

3.2.2.1 Mise en place d'un filtre

Le profil par défaut génère un graphique monochrome, où tous les protocoles sont mélangés. L'objectif de notre filtre sera de séparer les différents protocoles.

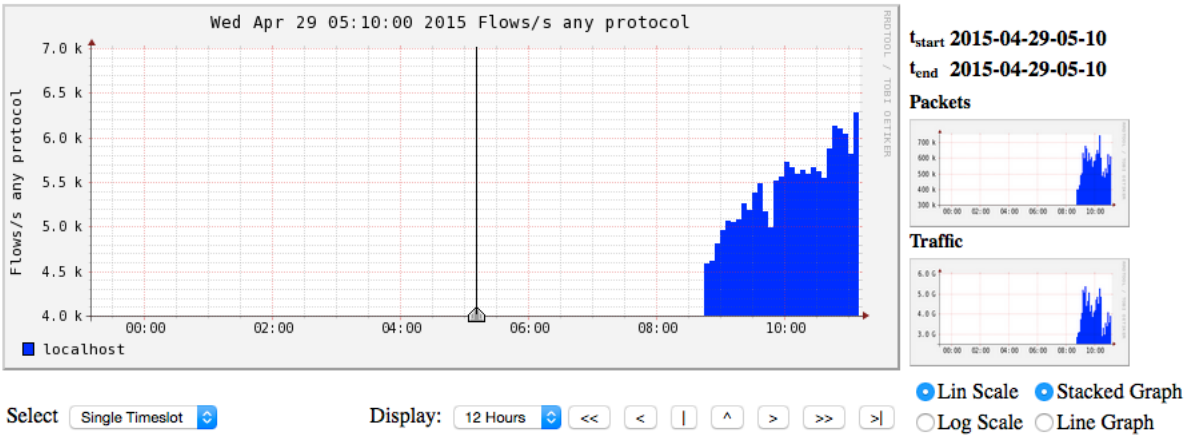


FIGURE 26 – NFSen : Graphe sans filtre

La première étape est la création d'un nouveau profil. Au lieu d'avoir un seul canal, nous créons un canal par protocole. Ce qui permet de les séparer. Le choix des protocoles est libre. Nous avons fait le choix ici d'effectuer le filtre sur toutes les IP. Nous aurions aussi pour le faire par sous-réseau.

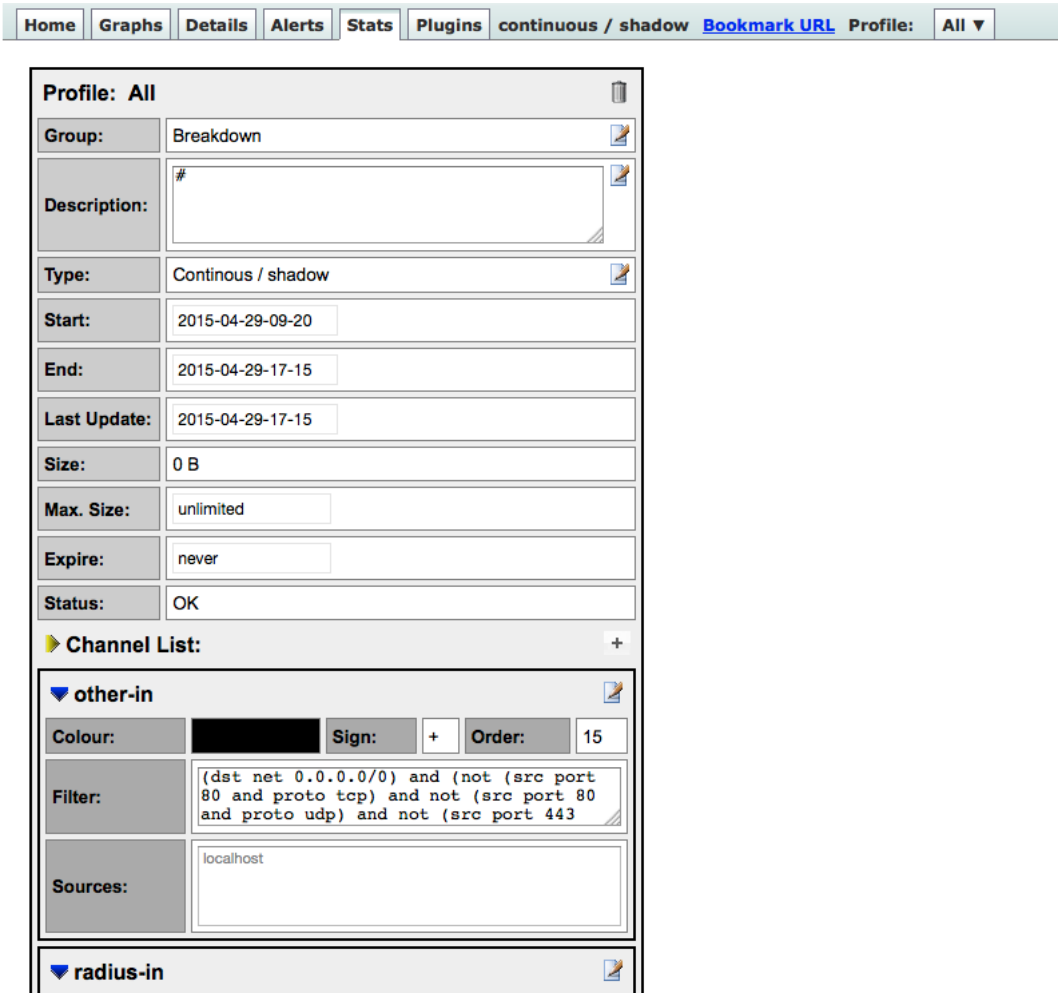
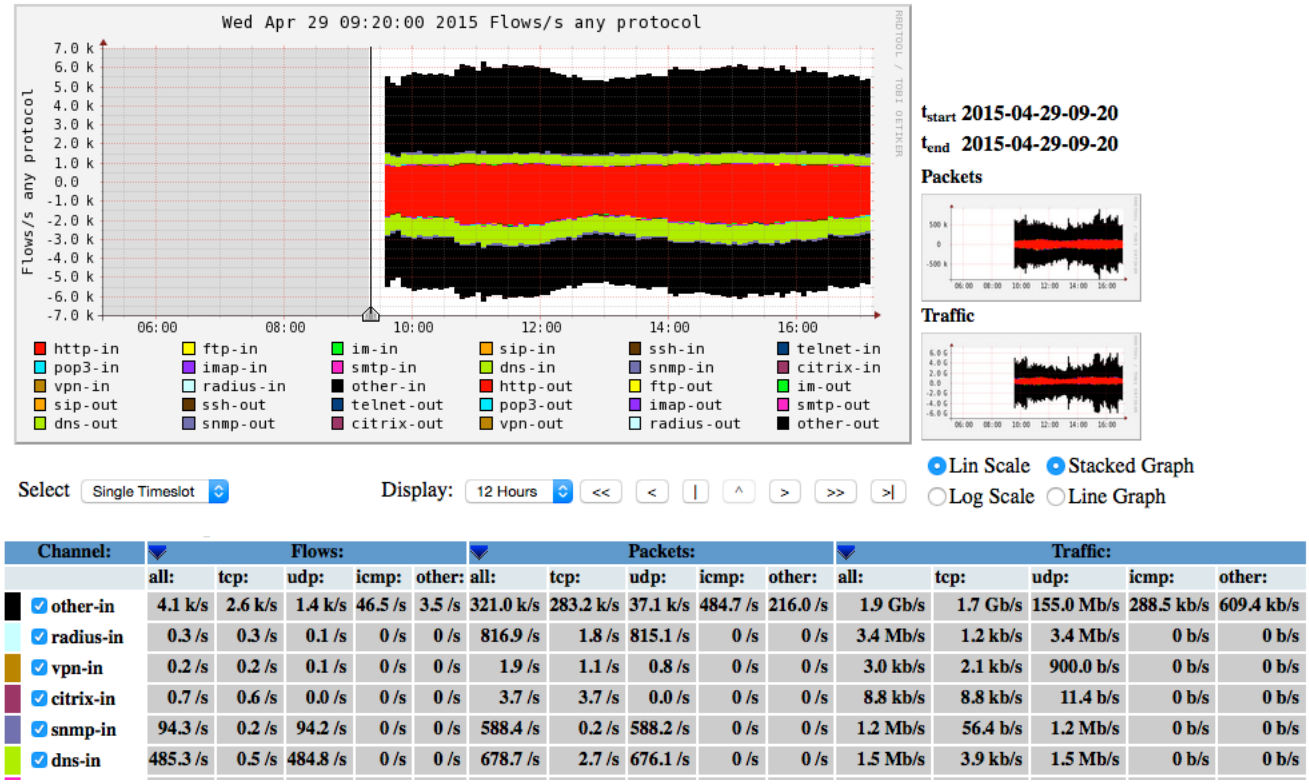


FIGURE 27 – NFSen : Création profil

Une fois ce profil sélectionné, nous pouvons voir que les différents protocoles sont séparés. Pour chaque protocole nous avons le trafic entrant (partie positive du graphique) et sortant (partie négative du graphique). Nous avons également la possibilité de ne pas faire apparaître certains protocoles, simplement en les décochant.



Channel:	Flows:					Packets:					Traffic:				
	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:	all:	tcp:	udp:	icmp:	other:
other-in	4.1 k/s	2.6 k/s	1.4 k/s	46.5 /s	3.5 /s	321.0 k/s	283.2 k/s	37.1 k/s	484.7 /s	216.0 /s	1.9 Gb/s	1.7 Gb/s	155.0 Mb/s	288.5 kb/s	609.4 kb/s
radius-in	0.3 /s	0.3 /s	0.1 /s	0 /s	0 /s	816.9 /s	1.8 /s	815.1 /s	0 /s	0 /s	3.4 Mb/s	1.2 kb/s	3.4 Mb/s	0 b/s	0 b/s
vpn-in	0.2 /s	0.2 /s	0.1 /s	0 /s	0 /s	1.9 /s	1.1 /s	0.8 /s	0 /s	0 /s	3.0 kb/s	2.1 kb/s	900.0 b/s	0 b/s	0 b/s
citrix-in	0.7 /s	0.6 /s	0.0 /s	0 /s	0 /s	3.7 /s	3.7 /s	0.0 /s	0 /s	0 /s	8.8 kb/s	8.8 kb/s	11.4 b/s	0 b/s	0 b/s
snmp-in	94.3 /s	0.2 /s	94.2 /s	0 /s	0 /s	588.4 /s	0.2 /s	588.2 /s	0 /s	0 /s	1.2 Mb/s	56.4 b/s	1.2 Mb/s	0 b/s	0 b/s
dns-in	485.3 /s	0.5 /s	484.8 /s	0 /s	0 /s	678.7 /s	2.7 /s	676.1 /s	0 /s	0 /s	1.5 Mb/s	3.9 kb/s	1.5 Mb/s	0 b/s	0 b/s

FIGURE 28 – NFSen : Graphe avec filtre

La mise en place de ce filtre nous permet de montrer qu'il est assez simple de pouvoir dégager des informations à partir de données peu lisibles à la base. Une infinité de filtres pourrait être imaginée. Le but ici n'est pas de montrer toutes les possibilités mais de montrer la façon de les mettre en place.

3.2.2.2 Configuration d'une alerte

Dans l'onglet "Alerts", nous avons la possibilité de configurer des alertes en fonction de conditions. Pour qu'une alerte soit levée, les conditions doivent être remplies. Nous avons la possibilité d'envoyer un mail pour nous avertir qu'une alerte est levée.

L'alerte seule n'a pas beaucoup d'intérêt dans la plupart des cas car bien qu'on soit averti qu'un seuil a été atteint, nous ne savons pas par qui. Une analyse doit ensuite être effectuée pour le savoir. Les plugins permettent d'améliorer cet aspect.

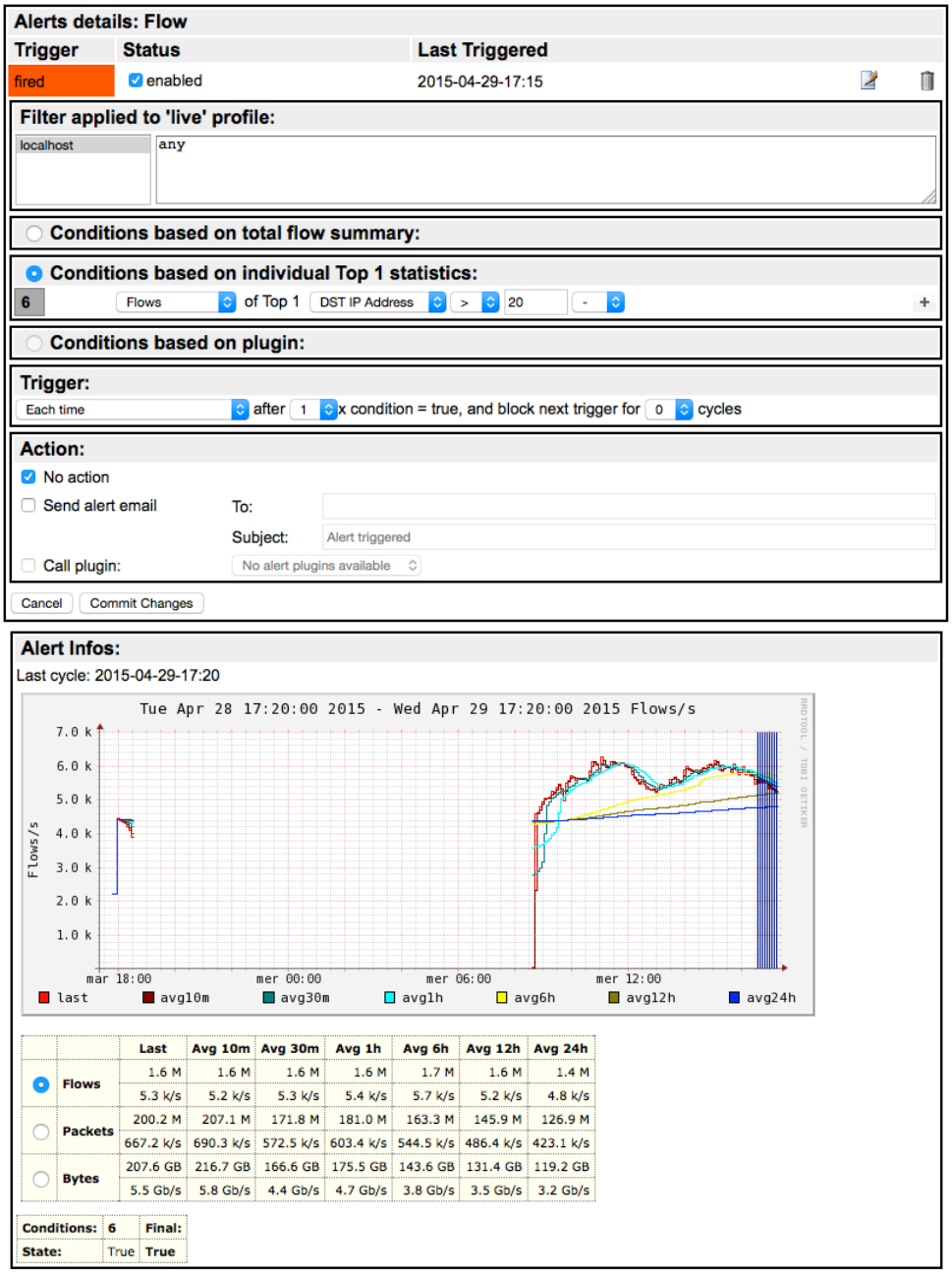


FIGURE 29 – NFSen : Création d'alerte

3.2.2.3 Mise en place d'un plugin

Les plugins se composent de deux parties, un backend et un frontend. Le backend est un module Perl et est chargé au démarrage de NFSen. C'est lui qui traite et analyse les données, qui contient les conditions d'alerte et les actions à prendre en cas d'alerte. Le frontend est écrit en PHP. Son rôle est d'afficher les résultats du backend ou d'entrer des données qui seront utilisées par lui. On peut considérer que le frontend est l'interface graphique du backend [nfsen.net] [10].

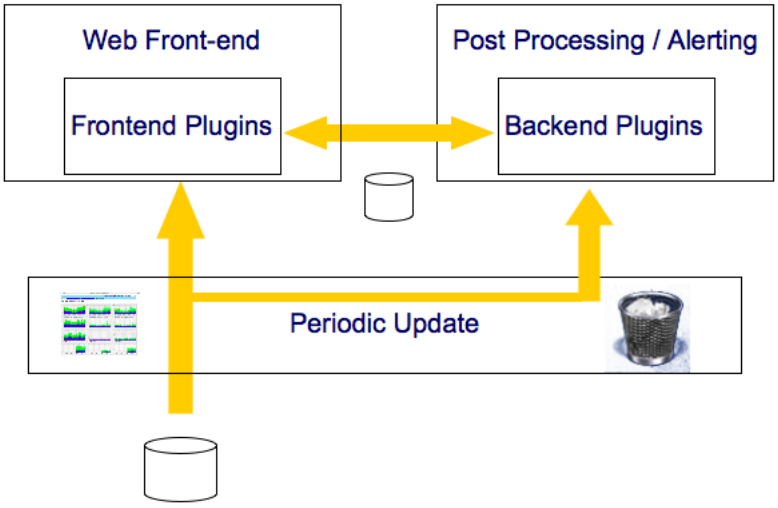


FIGURE 30 – NFSen : Concept des plugins [sourceforge.net]

Par faute de temps, il ne nous a pas été possible de développer notre propre plugin. Pour illustrer le concept nous avons choisi de mettre en place un plugin développé par la faculté d'informatique de l'Université Masaryk en République Tchèque. Le plugin s'appelle "cndet" [6]. Il détecte des malwares en utilisant plusieurs patterns de détection basés sur le comportement de botnets. Le plugin ici détecte le botnet "Chuck Norris". La particularité de ce botnet est qu'il vise les modems et routeurs DSL et non des ordinateurs. La chance de détecter ce botnet dans notre réseau est donc nul mais le but est d'illustrer le concept des plugins.

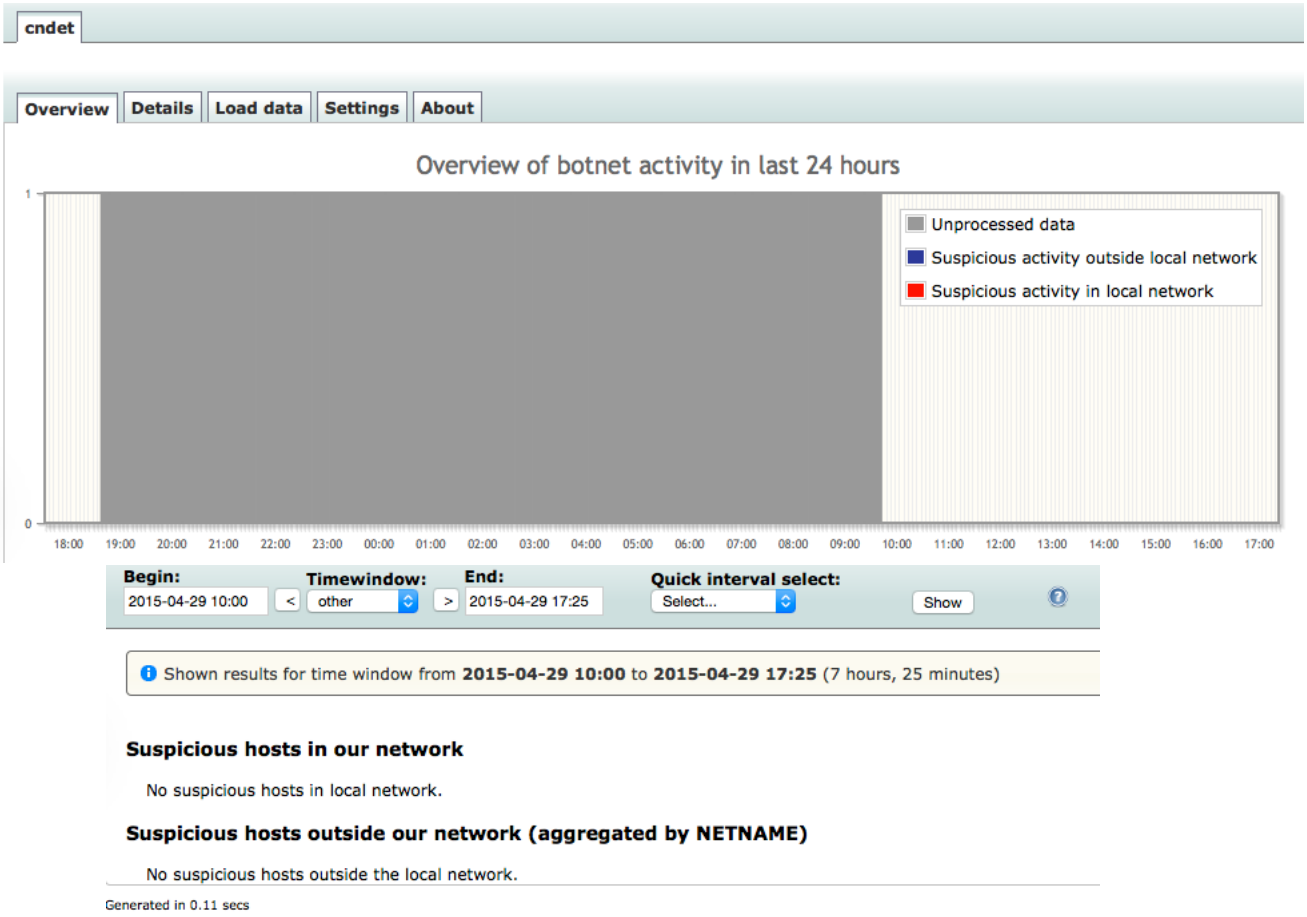


FIGURE 31 – NFSen : plugin cndet

D'autres plugins sont disponibles et permettent d'ajouter des fonctionnalités intéressantes à NFSEN

Voici quelques exemples :

- **InternetAlerts** (Csnet) : utilise des listes d'ordinateurs infectés, de serveur de spam, phishing, bonnets, etc. Ces listes sont fournies par deux systèmes d'alerte en ligne (AMaDa project et Shadowserver) ;
- **HostStats** (Csnet) : il calcule des statistiques pour chaque adresse IP, ces statistiques sont ensuite utilisées pour découvrir des adresse IP suspectes ;
- **SURFmap** (University of Twente) : permet de fournir des informations géographiques sur le trafic réseau en utilisant l'API Google Maps ;
- **SSHCure** (University of Twente) : permet de détecter les attaques SSH. Il peut faire la différence entre une attaque réussie et une attaque qui a échoué ;
- **PortTracker** (Masaryk University) : permet de monitorer les ports.

Nous pourrions imaginer développer un plugin qui détecte les botnet de façon plus générale. Si on se base sur le fait qu'un botnet se connecte à un contrôleur, une fois l'adresse de ce contrôleur connue nous pouvons le rajouter à notre liste. Une liste de contrôleurs connus peut être téléchargée sur divers sites (ex : https ://zeustracker.abuse.ch/). Le plugin nous enverra un message d'alerte via mail pour nous prévenir qu'un host communique avec un serveur et est donc probablement infecté.

Nous pourrions aussi créer un plugin pour détecter les P2P en se basant sur leurs caractéristiques. Le trafic descendant d'un P2P est caractérisé par un nombre important de sources se connectant à une même IP de destination sur un port spécifique (les ports sources peuvent varier). Le trafic montant a des caractéristiques inverses.

Afin de rendre ce filtre plus efficace, nous pourrions nous baser sur la thèse de Ahmed Bashir de l'Université Carleton dont le sujet est "Classifying P2P Activities in Netflow Records : A Case Study (BitTorrent & Skype) [15]". Son étude consiste à identifier les applications P2P à travers les NetFlow de façon fiable. Comme déjà mentionné plus haut, l'auteur insiste sur le fait que le modèle (filtre) doit être mis jour lorsque des mises à jour majeures des applications sont publiées, afin de maintenir la précision et d'éviter une mauvaise identification de l'application qui est détectée.

Nous pouvons constater que les possibilités offertes par les plugins sont nombreuses. L'outil NFSen qui, à la base, n'offre pas beaucoup de fonctionnalités peut être transformé en un outil très puissant. En combinant les filtres, alertes et plugins, une infinité de fonctionnalités peuvent lui être ajoutées.

A défaut d'avoir pu développer nos propres plugins, nous allons imaginer quelques scripts dans la partie suivante. Ces scripts permettront de montrer ce qu'il est possible de détecter à l'aide des traces NetFlow.

3.2.3 Réalisation de scripts

Dans cette partie nous allons réaliser des scripts permettant d'analyser les traces NetFlow, afin de détecter des anomalies. Le langage choisi est le Perl, dans le but de faciliter la réutilisation de ces scripts pour en faire des plugins NFSen. Les plugins NFSen sont écrits en Perl. Nous utiliserons l'API NFDump pour traiter les fichiers.

Le premier script permettra d'obtenir un graphique du nombre de destinations par rapport au volume échangé pour chaque poste d'un réseau déterminé. Il permettra donc de comparer les postes entre eux.

Les trois scripts suivants permettront de visualiser l'évolution du nombre de destinations contactées, le volume échangé et les nombres de flux d'un poste au cours du temps.

Le cinquième script permettra d'obtenir un graphique représentant les fréquences des échanges entre un poste et une destination déterminée. Le but sera de repérer les échanges ayant une certaine périodicité.

Pour tester nos scripts, notre réseau de test étant découpé en VLAN, nous avons choisi de nous occuper d'un VLAN à la fois. Dans un VLAN, les postes présents doivent normalement avoir le même type d'utilisation. Leur profil doit donc être quasi identique.

Nous avons choisi deux VLAN dans lesquels nous savons que des postes sont infectés. L'objectif est donc de les repérer en détectant des anomalies provenant de ces postes à l'aide des scripts.

3.2.3.1 Script 1 : Nombre de destinations vs Volume (voir code source Annexe A p.69)

Le script génère un graphique du nombre de destinations par rapport aux volumes échangés pour chaque poste du VLAN.

Le script peut porter sur un jour ou un mois. Une fonction permet de ne prendre en compte que les jours ouvrables et exclure les périodes de vacances et les weekends. Dans les exemples ci-dessous, nous n'avons ni exclu de jours, ni spécifié de plage horaire.

Les postes ayant un nombre de destinations et un volume de données sensiblement plus importants seront considérés comme suspects.

3.2.3.2 Script 2 : Evolution du nombre de destinations (voir code source Annexe B p.75)

Le script génère un graphique permettant d'observer l'évolution du nombre de destinations contactées par un poste déterminé au cours du temps. Nous comparons donc le comportement du poste avec lui même, jour par jour. Un trop grand écart sera considéré comme anormal.

3.2.3.3 Script 3 : Evolution du volume de données échangées (voir code source Annexe B p.75)

Le script génère un graphique permettant d'observer l'évolution du volume de données échangées pour un poste déterminé au cours du temps. Nous comparons donc le comportement du poste avec lui même, jour par jour. Un trop grand écart sera considéré comme anormal.

3.2.3.4 Script 4 : Evolution du nombre de flux (voir code source Annexe C p.80)

Le script génère un graphique permettant d'observer l'évolution du nombre de flux générés par un poste déterminé au cours du temps. Nous comparons donc le comportement du poste avec lui même, jour par jour. Un trop grand écart sera considéré comme anormal.

3.2.3.5 Script 5 : Analyse des fréquences des échanges entre deux hosts (voir code source Annexe D p.84)

Le script génère un graphique pour chaque paire IP source <=> IP destination + port destination. Il met en évidence les fréquences des échanges en indiquant l'amplitude de celles-ci. Plus l'amplitude d'une fréquence est grande plus cette fréquence se retrouve dans les échanges. Si l'amplitude d'une fréquence est beaucoup plus grande que celle des autres fréquences qui ont une amplitude quasi-nulle, on peut considérer que l'échange est périodique.

L'objectif est donc d'analyser le signal temporel dans l'espace fréquentiel. Pour réaliser cette analyse, nous ferons appel à la transformée de Fourier discrète.

Nous devons échantillonner notre signal pour pouvoir calculer la transformée de Fourier. Lorsqu'une communication existe au moment de l'échantillon, un 1 est inscrit, 0 dans le cas contraire. Une règle à respecter est que la fréquence d'échantillonnage doit être supérieure au double de la fréquence analysée (Shannon). Pour respecter cela, un premier calcul est effectué sur le signal afin de déterminer la période la plus courte entre deux échanges d'un couple IP source <=> IP destination + port destination. La fréquence d'échantillonnage sera le quart de cette période.

Une fois la fréquence d'échantillonnage déterminée, l'échantillonnage est effectué. Le signal est donc transformé en une suite de 0 et de 1.

Cette suite binaire est alors analysée à l'aide de la transformée de Fourier discrète et un graphe des fréquence est généré.

Le script est encore à ce stade à l'état de prototype. Des améliorations et optimisations doivent lui être apportées. En effet, lorsqu'un poste communique avec un grand nombre de destinations différentes, les temps de calcul explosent et l'analyse devient problématique. Etant donné qu'un graphe est généré par couple IP source <=> IP destination + port destination, il faut ensuite les analyser manuellement. Afin de réduire le nombre de graphes, une première amélioration a été apportée en excluant les communications ayant une durée assez courte (ex : n'apparaissent qu'une seule fois). Il faudrait idéalement que le script puisse déterminer automatiquement les signaux périodiques et ne générer un graphe que dans ce cas.

3.2.3.6 Détection d'anomalies à l'aide des scripts - scénario 1

Dans ce scénario nous savons qu'un certain nombre de postes font partie d'un botnet. Nous allons tenter de les identifier à l'aide de nos scripts.

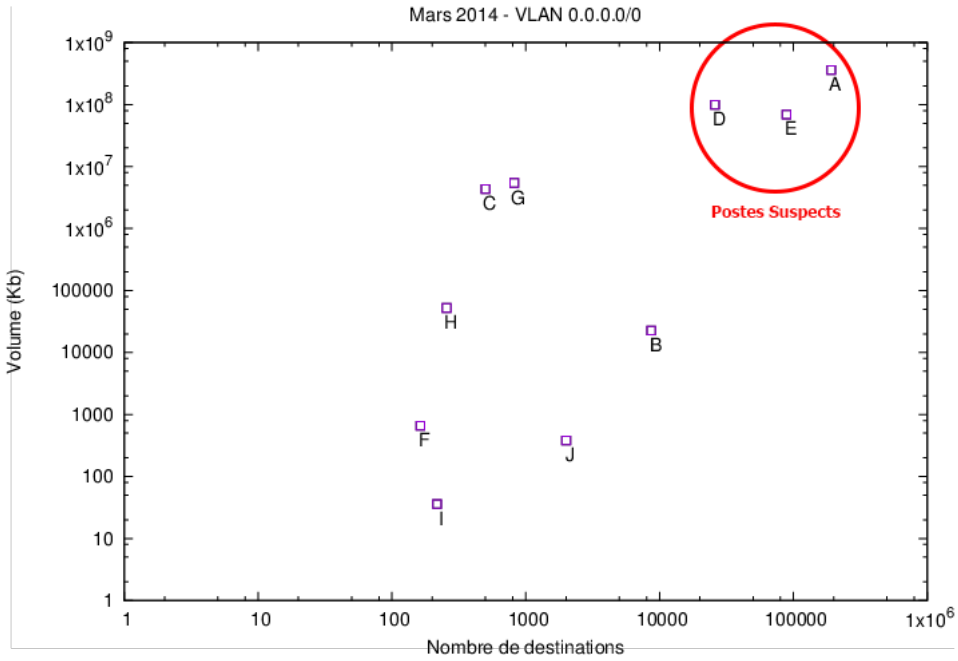


FIGURE 32 – Script 1 : scénario 1

Les adresses IP des postes représentés dans le graphique ci-dessus (figure 32) ont été remplacées par des lettres pour des raisons de confidentialité.

Nous pouvons constater que les hosts A,D,E ont un nombre de destinations et un volume d'échanges plus importants que les autres. Ils sont donc considérés comme "suspects".

Intéressons nous de plus près au poste D, afin de confirmer son statut de poste "suspect".

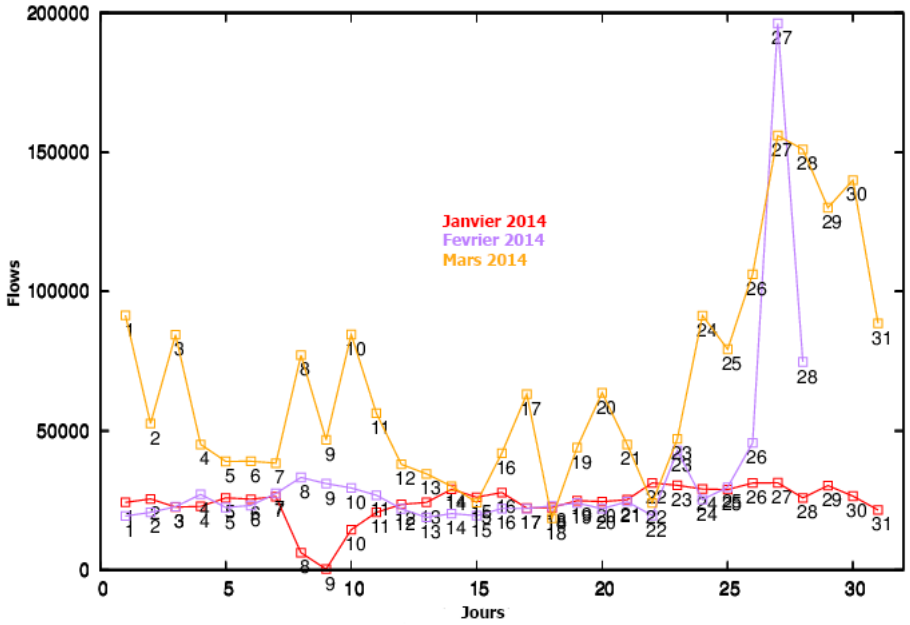


FIGURE 33 – Script 4 : scénario 1

Nous pouvons constater sur le graphique de la figure 33, que le nombre de flux augmente subitement fin février. Cette augmentation continue durant le mois de mars.

Ce graphe confirme l'anomalie relevée à l'aide du script 1.

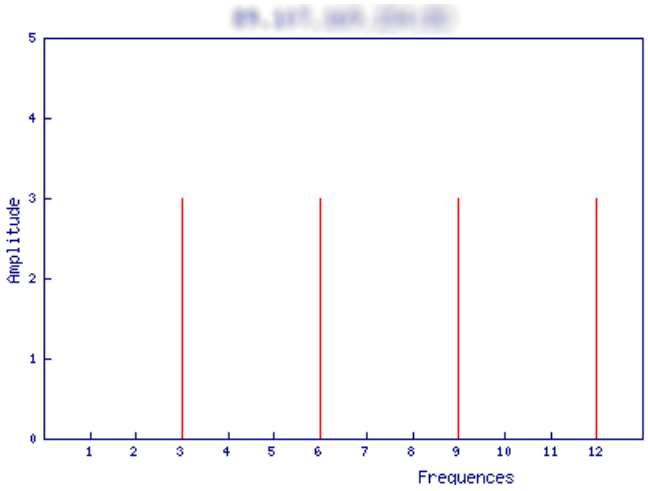


FIGURE 34 – Script 5 : scénario 1

Nous avons voulu tester le script 5, en étant conscient qu'il est toujours à l'état expérimental. Il nous a permis de repérer une communication périodique provenant de notre poste "suspect". Certains membres d'un botnet communiquent à interval régulier avec leur serveur C&C.

Il est important de noter qu'un trafic périodique n'est pas automatiquement anormal. De nombreux services et sites web génèrent du trafic périodique. Dans notre cas, il s'avère fortuitement que c'est le seul trafic périodique qui a été trouvé sur ce poste-là.

Les traces analysées datant 2013, nous n'avons pas pu vérifier notre suspicion directement sur le poste. Ce n'est pas vraiment un problème étant donné que cette infection a été détectée à l'époque. Nous avons donc pu vérifier notre analyse et nos soupçons.

Les postes A,C,D,G faisaient partie du botnet. Les postes A et D apparaissent bien comme suspects sur le graphe de la figure 32. Le poste E est un faux positif. Ce faux positif peut être expliqué par le fait que les postes de ce scénario ne sont pas des postes classiques mais des serveurs. Le traffic généré peut donc varier fortement d'un poste à l'autre et générer des faux positifs.

Les postes C et G n'apparaissent pas comme suspects car n'étaient pas en activité à ce moment-là.

La communication mise en évidence par le script 5 à la figure 34 avait pour destination un serveur C&C du botnet observé. Les postes A,C,D,G communiquaient tous les 4 avec ce serveur. Une recherche sur cette destination nous aurait permis de voir que les 4 postes étaient contaminés.

3.2.3.7 Détection d'anomalies à l'aide des scripts - scénario 2

Dans ce deuxième scénario nous savons qu'un poste est compromis et sert à effectuer des attaques (une plainte a été reçue concernant ce poste, en nous signalant qu'il était l'auteur d'attaques). Voyons si nos scripts nous permettent d'identifier ce poste. Dans

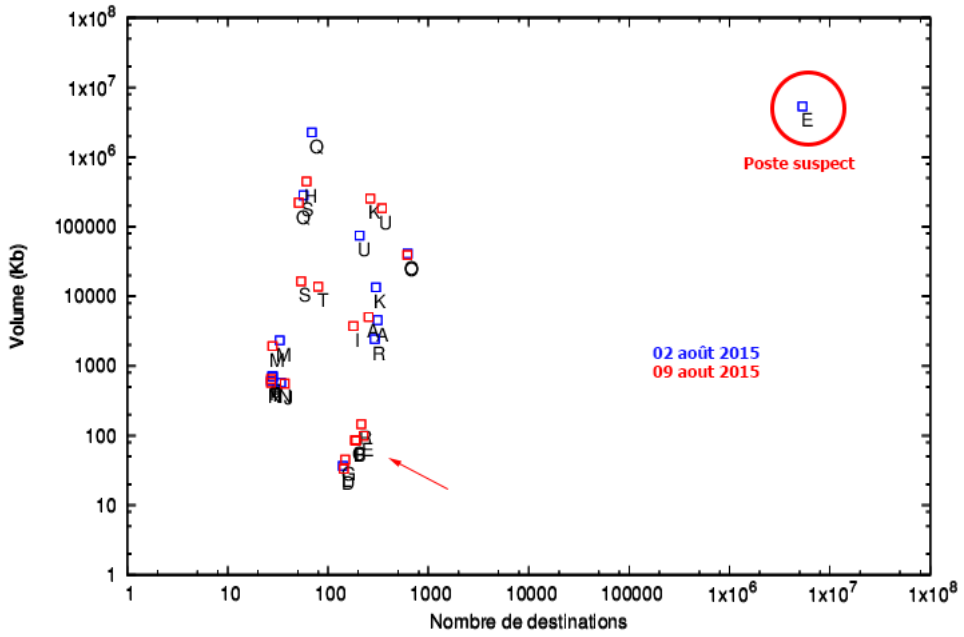


FIGURE 35 – Script 1 : scénario 2

le graphique de la figure 35, nous pouvons constater que le poste E est "suspect". Son volume de données échangées et son nombre de destinations sont beaucoup plus élevés que les autres postes de ce VLAN. Le poste E a bien comme IP, l'IP pour laquelle nous avons reçu une plainte. Le script 1 nous a donc permis d'identifier le poste que l'on cherchait.

Les scripts nous ont permis d'identifier des postes infectés et faisant partie d'un botnet. Ces scripts sont basés sur l'analyse du comportement des postes, afin de détecter des anomalies.

Dans le cas du script 1, les postes sont comparés entre eux. Les postes s'écartant trop du profil général sont considérés comme suspects.

Dans les scripts 2,3,4, le poste est comparé avec lui même. Un poste ayant une trop grande déviation par rapport à son comportment habituel est considéré comme suspect.

Pour un maximum d'efficacité, il est donc nécessaire que les postes comparés aient des utilisations similaires. En résumé nous ne pouvons comparer que ce qui est comparable.

4 Conclusion

Nous voilà arrivés au terme de notre étude et de notre réflexion au sujet du post traitement des traces NetFlow. Notre démarche a-t-elle permis de répondre à notre question de départ qui était : "Que faut-il mettre en œuvre pour appliquer la politique de sécurité réseau de l'UCL, qui découle de ses codes d'éthique et de déontologie ?"

Afin de répondre à cette question, nous avons suivi une méthodologie qui nous a amené à nous poser un certain nombre de nouvelles questions auxquelles nous avons répondu.

La capture des traces NetFlow et l'analyse de celles-ci est-elle légale ? L'étude de la législation applicable en la matière, nous a permis de répondre par l'affirmative.

Notre démarche est basée sur les NetFlow. Si nous avions eu le choix entre plusieurs technologies aurions-nous choisi les NetFlow ? Une comparaison avec d'autres technologies a montré que notre choix était judicieux. L'analyse des enregistrements NetFlow est facile à mettre en place et peu coûteuse.

Après avoir répondu à ces deux questions, nous avons étudié le protocole NetFlow. Le protocole est assez simple mais malgré sa simplicité il nous fournit néanmoins une quantité importante d'informations telles que le volume échangé, qui communique avec qui, le nombre de connexions, etc.

D'autres questions sont alors apparues :

Comment détecter des activités anormales à l'aide des traces NetFlow ? A cette fin, nous avons étudié différentes techniques. La première était de comparer le trafic réseau à un modèle reflétant le trafic normal. Tout écart de ce modèle peut être considéré comme anormal. La seconde était basée sur des pattern d'attaques. Une fois qu'une attaque est identifiée, nous pouvons analyser son comportement afin de construire un pattern. Tout trafic correspondant à ce pattern peut être considéré comme anormal.

Comment un système d'analyse NetFlow doit-il être mis en place ? Pour répondre à cette question, nous avons étudié la façon dont un système d'analyse NetFlow devait être mis en place en suivant des recommandations.

Notre étude théorique étant terminée, nous sommes passé à la mise en pratique des concepts abordés.

Quel produit d'analyse choisir parmi l'offre assez large et surtout à partir de quels critères ? Nous avons mis en place une méthodologie afin d'évaluer les différents produits

permettant l'analyse d'enregistrements NetFlow. Nous l'avons ensuite appliquée à notre contexte.

La méthodologie nous a permis d'effectuer une première sélection basée sur les exigences des stakeholders. Une fois ce premier choix effectué, nous les avons comparés afin de déterminer quel produit répondait le mieux techniquement aux exigences. Le problème majeur a été de trouver des données de test. Nous voulions que les données soient identiques pour tous les tests. Plusieurs solutions ont été envisagées. La solution retenue a été de rejouer des NetFlow capturés au préalable.

Pour la comparaison finale nous avons rencontré deux autres problèmes.
Le premier concernait, à nouveau, les données de test. Nous avions des données mais nous étions incapables de savoir ce qu'elles contenaient. Nous entendons par là que nous ne savions pas si des activités anormales étaient présentes ou non.

Le second problème était que les différents produits retenus avaient parfois des fonctionnalités que les autres produits n'offraient pas ou du moins pas nativement.

Notre idée de base était de tester les trois produits retenus afin de vérifier qu'ils détectaient les mêmes anomalies. Compte tenu des deux problèmes rencontrés, nous n'avons pas pu effectuer ces tests. Il nous aurait fallu beaucoup trop de temps pour adapter chaque produit à nos besoins. Nous avons alors décidé d'effectuer des tests "génériques". Ces tests ont eu pour but de vérifier que les 3 produits nous renvoyaient les mêmes résultats. Les différents tests ont été choisis en fonction du type d'informations recherchées lors de la détection d'anomalies. Les 3 produits ont donné les mêmes résultats aux tests.

Pour effectuer le choix final nous avons donc regardé quel était le produit le plus flexible nous permettant d'ajouter les fonctionnalités répondant au mieux à nos exigences. Le seul produit qui nous le permettait était NFSen. C'est pour cette raison que nous l'avons choisi.

Une fois le choix effectué, nous l'avons mis en production. Nous avons commencé le développement de nouvelles fonctionnalités et en avons imaginé d'autres comme la détection et la classification du trafic P2P. L'outil est en place et fonctionnel. Il faut maintenant imaginer de nouvelles fonctionnalités et les développer. Les possibilités sont grandes.

Afin de démontrer l'utilité et les possibilités de détection des traces NetFlow, nous avons développé des scripts. Ces scripts nous ont permis de détecter des anomalies. Nous avons aussi vu l'importance de comparer des postes ayant un profil similaire ou un poste avec lui même. Sans cela les résultats peuvent être faussés.

Pour conclure, nous pouvons dire que l'analyse et la mise en place d'outils destinés à l'exploitation des données NetFlow permettent en partie d'appliquer la politique de sécurité réseau. En effet, elles nous permettent de détecter des anomalies sur le réseau. Ces anomalies peuvent être générées par des malwares, des botnets, un usage inapproprié du réseau (exemple : téléchargement d'œuvres protégées par droits d'auteurs), du spam, etc.

Il répond en partie à notre question de départ car comme nous l'avons mentionné dans le mémoire, la sécurité doit être en couche et remise sans cesse en question. Le protocole

nous permet de détecter rapidement et facilement un certain nombre de menaces. Si, par exemple, la menace n'est pas active, elle ne créera pas d'anomalie au niveau du trafic réseau. Nous ne détecterons pas non plus les menaces ne générant pas une variation significative du trafic provenant du poste analysé. D'autres outils et techniques (DPI, antivirus, analyse du poste, etc.) doivent ensuite êtres utilisés pour affiner l'analyse afin de déterminer l'origine de l'anomalie ou permettre de détecter les menaces n'ayant pas créé d'anomalies au niveau du réseau. L'analyse des traces NetFlow nous permettra aussi dans certains cas de détecter des failles zero-day lorsque celle-ci est basée sur le comportement contrairement à des technologies basées sur les signatures.

NetFlow est un protocole simple, peu coûteux et facile à mettre en place mais une fois exploité, c'est un outil très puissant.

Bibliographie

[1] http ://en.wikipedia.org/wiki/netflow-netflow-and-ipfix.

[2] http ://en.wikipedia.org/wiki/netflow-netflow-equivalents.

[3] http ://fr.wikipedia.org/wiki/composant-pris-sur-etagere.

[4] http ://fr.wikipedia.org/wiki/netflow.

[5] https ://www.manageengine.com/products/netflow/.

[6] http ://www.muni.cz/fi/research/projects/4622/web/chuck-norris-botnet.

[7] http ://www.ntop.org/products/ntop/.

[8] http ://www.privacycommission.be/fr/la-surveillance-electronique-internet-email.

[9] nfdump.sourceforge.net.

[10] nfsen.sourceforge.net.

[11] Loi du 8 décembre 1992 relative à la protection de la vie privée à l'égard des traitements de données à caractère personnel. *M.B., 18 mars 1993*, 1992.

[12] Convention collective du travail n °81 du 26 avril 2002 relative à la protection de la vie privée des travaileurs à l'égard du contrôle de données de communication électroniques en réseau. *Commission vie privée*, 2002.

[13] Loi du 13 juin 2005 relative aux communications électroniques. *M.B., 20 juin 2005*, 2005.

[14] C.j.c.e. 24 novembre 2011(scarlet extended sa contre société belges des auteurs, compositeurs et éditeurs scrl (sabam)), c-70/10. *Rec. C.J.C.E., p.I-11959*, 2011.

[15] Ahmed Bashir. Classifying p2p activities in netflow records : A case study (bittorrnet and skype), 2012.

[16] Earl Carter. Intrusion detection systems. *Cisco Press*, 2002.

[17] Cisco. Netflow services solutions guide. *Cisco.com*, 2007.

[18] USmax corporation. Usmax methodology to conduct commercial off-the-shelf (cots) production evaluation. 2010.

[19] Justin Ellingwood. An introduction to snmp (simple network management protocol). *Digital Ocean*, 2014.

[20] Reynald Fléchaux. Jérémy d'hoinne, gartner : "pour le sdn, la sécurité n'est pas prête". *Silicon.fr*, Septembre 2014.

[21] Yiming Gong. Detecting worms and abnormal activities with netflow. *Securityfocus.com & symantec.com*, Novembre 2010.

[22] Pierre-Alain Goupille. *Technologie des ordinateurs et des réseaux*. Dundod, France, 2008.

[23] Yogita Hande, Aishwarya Jadhav, Achaleshwari Patil, and Rutuja Zagade. Software defined networking with intrusion detection system. *International Journal of Engineering and Technical Research (IJETR)*, Octobre 2014.

[24] Ivan Ivanovic. Recommendations for network traffic - analysis using the netflow protocol (best practice document) by amres nms group. *Geant.net*, Novembre 2011.

[25] Clément Lauriat. Introduction au monde de netflow. *J3TEL*, 2012.

[26] Bingdong Li, Jeff Springer, George Bebis Mehmet, and Hadi Gunes. A survey of network flow applications. *Journal of Network and Computer Applications*, Janvier 2013.

[27] Roberto Di Pietro and Luigi Mancini. *Intrusion Detection Systems*. Springer, Rome, 2008.

[28] Emmanuel S. Pilli, R.C. Joshi, and Radjdeep Niyogi. Network forensic frameworks : Survey and research challenges. *Elsevier*, 2010.

[29] Chris Smithee. Is openflow a new form of netflow ? *Lanscope.com*, 2012.

[30] Jean-Pierre Soulès. Comprendre le sdn, ou l'art de virtualiser le réseau. *Clubic Pro*, Avril 2014.

Annexes

Annexe A : Script 1 - nombre de destination vs volume

```
#! /usr/bin/perl

use feature "switch";
use Net::NfDump qw ':all';
use Data::Dumper;
use Config::Simple;
use DBI;
use Date::Business;
use GD::Graph::points;
use GD::Graph::Data;
use Chart::Gnuplot;

#Flow creation (filtering on fields srcip, dstip, bytes)
sub createFlow {

 my ($files,$subNet)=@_;
 my @array=@$files;
 printf "Create Flow...\n";
 $flow = new Net::NfDump(
  InputFiles => [@array],
                Filter => 'src net '.$subNet,
                Fields => 'srcip, dstip, bytes'
        );
        $flow->query();
 printf "Flow creation completed !\n";
        return $flow;
}
#Return month length
sub monthLength{

        my $l;
        my($month)=@_;

        given($month){
                when ("01") {$l=31;}
                when ("02") {$l=28;}
                when ("03") {$l=31;}
                when ("04") {$l=30;}
                when ("05") {$l=31;}
                when ("06") {$l=30;}
                when ("07") {$l=31;}
                when ("08") {$l=31;}
```

```
                when ("09") {$l=30;}
                when ("10") {$l=31;}
                when ("11") {$l=30;}
                when ("12") {$l=31;}
        }

 return $l;
}
#Return list of nfcapd files for a specific date
sub GetFilesList
{
 my ($j,$m,$a) = @_;
 $strJou=sprintf("%02d",$j);
        $strMoi=sprintf("%02d",$m);
 my $Path ="/Users/cluycx/Desktop/Data/".$a."/".$strMoi."/".$strJou."/";
        my $FileFound;
        my @FilesList=();

       # Read files list
        opendir (my $FhRep, $Path) or die "Impossible d'ouvrir le repertoire $Path\n";
        my @Contenu = grep { !/^\.\.?$/ } readdir($FhRep);
        closedir ($FhRep);

        foreach my $FileFound (@Contenu) {
        # Files processing
  if (( -f "$Path/$FileFound")&&(((substr $FileFound, 15,2)>=8)&&((substr $FileFound,
  15,2)<=17))) {
          push ( @FilesList, "$Path/$FileFound" );
         }
       # Directories processing
  elsif ( -d "$Path/$FileFound" ) {
  # Recursive search
   push (@FilesList, GetFilesList("$Path/$FileFound") );
                }
 }
 return sort @FilesList;
}
#Filling hash (keys : srcip) with specifics fields
sub firstGraph {

 my ($files,$subNet,$h)=@_;
        my $flow =createFlow2($files,$subNet);
        my $totalBytes=0;
        my $totalDst=0;

 while (my ($srcip,$dstip,$bytes) = $flow->fetchrow_array() ) {
                $h->{ip2txt($srcip)}{ip2txt($dstip)} +=$bytes;
        }
        $flow->finish();
}
#Read hash, calculate totals and call drawing graphs functions
sub hashToData {

 my ($h,$subNet,$mois,$annee)=@_;
 $subNet=~tr/./_/;
        $subNet=(split(/\//,$subNet))[0];
 my $i = 0;
 my @data=();
```

```
 my @data2=();
 my @data3=();
        my @data4=();
 my $name;
 foreach my $srcip (sort keys %{$h}) {
                $totalBytes=0;
                $totalDst=0;
                foreach my $dstip (keys %{ $h->{$srcip} }) {
   if (((substr $dstip, 0, 7)ne"130.104")&&((substr $dstip, 0, 7)ne"192.168")){
                                $totalBytes+=$h->{$srcip}{$dstip};
                                $totalDst+=1;
                        }
                }
  $data[0][$i] = $totalDst;
  $data3[$i] = $totalDst;
         $data[1][$i] = int($totalBytes/1000);
  $data4[$i] = int($totalBytes/1000);
  $data2[$i] = $srcip;
  $i++;
 }

 $name = $subNet."_".$mois."_".$annee;

#Sort \#destinations

 my @idx = sort { $data[0][$a] <=> $data[0][$b] } 0 .. $#{$data[0]};
 my @temp1;
 my @temp2;

 for (my $t = 0; $t<=($#{$data[0]}); $t++){
  #printf "--->%d %d\n",$data[0][$t],$#{$data[0]} ;
  $temp1[$t] = $data[0][$t];
  $temp2[$t] = $data[1][$t];
 }
 @temp1 = @temp1[@idx];
 @temp2 = @temp2[@idx];

#DataSet (\#destination/volume) pour graphmaker
 $data[0]=\@temp1;
 $data[1]=\@temp2;
#Label
 @data2=@data2[@idx];

#DataSet (\#destination) pour graphmaker2
 @data3=@data3[@idx];
#DataSet (volume) pour graphmaker2
 @data4=@data4[@idx];

 graphMaker(\@data,$name);
 graphMaker2(\@data3,\@data4,\@data2);

}
#Return business days
sub calcBusinessDays {

        my ($s, $e, @holidays) = @_;
        my @jours;
        print "@holidays\n";
```

```
        $start = new Date::Business(DATE => $s);
        $end = new Date::Business(DATE=> $e);

        $startHolidays = new Date::Business(DATE => @holidays[0]);
        $endHolidays = new Date::Business(DATE => @holidays[1]);

        printf "Start Holidays : %s \n", $startHolidays->image();
        printf "End Holidays : %s \n", $endHolidays->image();

        while (!$start->gt($end)) {
                if (($start->lt($startHolidays)==1)||($start->gt($endHolidays)==1)){
                        #printf "%s \n",substr $start->image(), 6, 8;
                        push @jours,substr $start->image(), 6, 8;
                }

                $start->nextb();
        }
        return @jours;
}
#Draw graph
sub graphMaker {

 my ($data,$name) = @_;


#  $values->set_y(1, 1, undef);
#  $values->set_y(2, 0, undef);

 my $graph = GD::Graph::points->new(800,1000);
 my $title = (split(/_/,$name))[0].":".(split(/_/,$name))[1];
 my $data2 = GD::Graph::Data->new($data);

 $graph->set(
      x_label         => 'Nbr. sources',
      y_label         => 'Volume',
      title           => $title,
      show_values => 0,
      #y_max_value     => 7,
      #y_tick_number   => 8,
      #y_label_skip    => 3,
  #x_labels_vertical => 1,
      marker_size        => 1,
      #bar_spacing     => 10,
      #shadow_depth    => 4,
      #shadowclr       => 'dred',
      #transparent     => 0,
      correct_width  => 0,
 ) or die $graph->error;

 #$graph->set_legend(@$data);
 #$graph->set(show_values => $values);
 $graph->plot($data2) or die $graph->error;
 $name =~tr/./_/;
 my $file = $name.".png";
 open(my $out, '>', $file) or die "Cannot open '$file' for write: $!";
 binmode $out;
 print $out $graph->gd->png;
```

```
 close $out;
}
#Draw graph
sub graphMaker2 {

 my ($x,$y,$labels)=@_;

 my $chart = Chart::Gnuplot->new(
         output => "simple.png",
         title  => "VLAN 0.0.0.0/26",
         xlabel => "Nombre de destinations",
         ylabel => "Volume (Mb)",
     );
 my $dataSet = Chart::Gnuplot::DataSet->new(
        xdata => $x,
        ydata => $y,
#title => "Plotting a line from Perl arrays",
        style => "points",
 pointtype =>'square',

);

 my $j;
 for (my $i=0;$i<=$#$labels;$i++){
  if ((@$y[$i])==0){
                        $j=1;
                }
                else {
                        $j=@$y[$i];
                }

  $chart->label (
   position => @$x[$i].",".$j." center",
          text    => @$labels[$i],
          offset => "0, -1,5",
          #pointcolor => 'dark-green',
          pointtype => 'square',
          #pointsize => 2,
  );
 }

 $chart->set(logscale=>'xy');
 $chart->plot2d($dataSet);
}

#Array srcip|dstip|bytes
my %hash;
printf « Démarrage de l'analyse\n »;

my @subnets = ('0.0.0.0/26');

#Holidays
my @holidays = ('20131224','20140101');
#Working days (- holidays)
my @jours = calcBusinessDays('20140101','20140131',@holidays);
#Mois traité
my $mois = 02;
my $annee = 2014;
```

```
foreach my $subNet(@subnets){
undef %hash;
 #foreach my $jour(@jours) {
 for (my $jour=1;$jour<2;$jour+=1) {
  printf "%d/%d/2013 for subnet : %s\n",$jour,$mois,$subNet;
  printf "-----------\n";
#Tableau contenant liste fichiers d'un jour
  @fileList = GetFilesList($jour,$mois,$annee);

  foreach my $File  (@fileList) {
   print "$File\n";
  }
  my $files = \@fileList;
#Remplissage tableau %hash jour par jour
  firstGraph2 ($files,$subNet,\%hash);
 }
print Dumper \%hash;
#Hash en graphique
hashToData (\%hash,$subNet,$mois,$annee);
}
```

Code 4 – Script 1 - nombre de destination vs volume

Annexe B : Script 2 et 3 - Evolution du nombre de destinations et du volume échangé

```
#! /usr/bin/perl

use feature "switch";
use Net::NfDump qw ':all';
use Data::Dumper;
use Chart::Gnuplot;

#Flow creation (filtering on fields srcip,dstip and bytes)
sub createFlow {

 my ($files,$subNet)=@_;
 my @array=@$files;
 printf "Create Flow...\n";
 $flow = new Net::NfDump(
   InputFiles => [@array],
                         Filter => 'src net '.$subNet,
                         Fields => 'srcip, dstip, bytes'
                         );
                         $flow->query();
 printf "Flow creation completed !\n";
                        return $flow;
}
#Return month length
sub monthLength{

        my $l;
        my($month)=@_;

        given($month){
                when ("01") {$l=31;}
                when ("02") {$l=28;}
                when ("03") {$l=31;}
                when ("04") {$l=30;}
                when ("05") {$l=31;}
                when ("06") {$l=30;}
                when ("07") {$l=31;}
                when ("08") {$l=31;}
                when ("09") {$l=30;}
                when ("10") {$l=31;}
                when ("11") {$l=30;}
                when ("12") {$l=31;}
        }

 return $l;
}
#Return list of nfcapd files for a specific date
sub GetFilesList
{
 my ($j,$m,$a) = @_;
 $strJou=sprintf("%02d",$j);
 $strMoi=sprintf("%02d",$m);
 my $Path ="/Volumes/VOL1/profiles-data/live/ucl/".$a."/".$strMoi."/".$strJou."/";
 my $FileFound;
```

```
 my @FilesList=();

        # Read files list
        opendir (my $FhRep, $Path) or die "Impossible d'ouvrir le repertoire $Path\n";
        my @Contenu = grep { !/^\.\.?$/ } readdir($FhRep);
        closedir ($FhRep);

        foreach my $FileFound (@Contenu) {
        # Files processing
         if ( -f "$Path/$FileFound"){

  push ( @FilesList, "$Path/$FileFound" );
         }
        # Directories processing
  elsif ( -d "$Path/$FileFound" ) {
  # Recursive search
   push (@FilesList, GetFilesList("$Path/$FileFound") );
  }
 }
 return sort @FilesList;
}

#Filling hash (keys : date) with specifics fields
sub firstGraph {

 my ($files,$subNet,$h,$date)=@_;
 my $flow =createFlow($files,$subNet);
 my $totalBytes=0;
 my $totalDst=0;
 while (my ($srcip,$dstip,$bytes) = $flow->fetchrow_array() ) {
  if (((ip2txt($srcip)) eq "0.104.0.0") ||
  ((ip2txt($srcip)) eq "0.0.0.0")){
   $h->{$date}{ip2txt($srcip)}{ip2txt($dstip)} +=$bytes;
  }
        }
        $flow->finish();
}

#Read hash, calculate totals and call drawing graphs functions
sub hashToData {

 my ($h,$subNet,$jour,$mois,$annee)=@_;
 $subNet=~tr/./_/;
        $subNet=(split(/\//,$subNet))[0];
 my $i = 0;
 my @data=();
 my @data2=();
 my @data3=();
 my $name;
 my $tabNbr=1;
 my %hash=%$h;
 foreach my $date (sort keys %hash) {
                $totalBytes=0;
                $totalDst=0;
  $totalBytes2=0;
  $totalDst2=0;
  $tabNbr=1;
 printf "Date : %s\n",$date;
```

```
  foreach my $srcip (keys %{$hash{$date}}) {
  printf "%s -> %s\n",$srcip,$date;

                foreach my $dstip (keys %{$hash{$date}{$srcip}}) {
    if (((substr $dstip, 0, 7)ne"130.104")&&((substr $dstip, 0, 7)ne"192.168")){
                                if ($srcip eq "0.0.0.0") {
      $totalBytes+=$hash{$date}{$srcip}{$dstip};
                                 $totalDst+=1;
                           }
     if ($srcip eq "0.0.0.0") {
      $totalBytes2+=$hash{$date}{$srcip}{$dstip};
                                                $totalDst2+=1;
     }
    }
                }
  $tabNbr++;
  }
 $data[$i] = $totalDst;
 $data2[$i] = int($totalBytes/1000);
 $data3[$i] = $date;
 $i++;
 }

$name = $subNet."_".$mois."_".$annee;

graphMaker(\@data,\@data2,\@data3,$subNet,$jour."_".$mois."_".$annee);

}
#Return an array with days of a month
sub calcMois {

 my ($mois) =@_;

 my $nbr = monthLength($mois);

 my @jours;

 for (my $i = 1;$i<=$nbr;$i++){
   if ($i<10)
  {
   push @jours,"0".$i ;
  }
  else {
   push @jours,"".$i ;
  }
 }

return sort @jours;
}

#Draw graph
sub graphMaker {

my ($x,$y,$labels,$subNet,$date)=@_;

my $chart = Chart::Gnuplot->new(
        output => "daily_".$subNet."_".$date.".png",
        title  => "VLAN ".$subNet,
```

```
        xlabel => "Nombre de destinations",
        ylabel => "Volume (Kb)",
    );
my $dataSet = Chart::Gnuplot::DataSet->new(
        xdata => $x,
        ydata => $y,
        #title => "Plotting a line from Perl arrays",
        style => "points",
 pointtype =>'square',

);

my $j;
for (my $i=0;$i<=$#$labels;$i++){
 if ((@$y[$i])==0){
                        $j=1;
                }
                else {
                        $j=@$y[$i];
                }

 $chart->label (
  position => @$x[$i].",".$j." center",
         text    => @$labels[$i],
         offset => "0, -1,5",
         #pointcolor => 'dark-green',
         pointtype => 'square',
         #pointsize => 2,
 );
}
 $chart->set(
  logscale=>'xy',
  xrange => [1, 1000000],
  yrange => [1, 1000000000],
 );
$chart->plot2d($dataSet);

}
printf "\\\\Analysing tool// \n";
printf "Subnet : %s\n",$subNet;
#Array srcip|dstip|bytes
my %hash;
printf "Start analyzing...\n";
my @subnets = ('0.0.0.0/26');
my $mois = "07";
my $annee = "2015";
my @jours = calcMois($mois);
print Dumper \@jours;
foreach my $subNet(@subnets){
undef %hash;
 foreach my $jour(@jours) {
  printf "%d/%d/2013 for subnet : %s\n",$jour,$mois,$subNet;
  printf "-----------\n";
#Array for days of a month
  @fileList = GetFilesList($jour,$mois,$annee);
  foreach my $File  (@fileList) {
   print "$File\n";
  }
```

```
 my $files = \@fileList;
#Filling hash day per day
 firstGraph ($files,$subNet,\%hash,$jour);
 }
hashToData (\%hash,$subNet,$jour,$mois,$annee);
}
```

Code 5 – Script 2 et 3 - Evolution du nombre de destinations et du volume échangé

Annexe C : Script 4 - Evolution du nombre de flux

```
#! /usr/bin/perl

use feature "switch";
use Net::NfDump qw ':all';
use Data::Dumper;
use Chart::Gnuplot;

#Flow creation (filtering on fields srcip,flows (aggreged))
sub createFlow {

        my ($files,$subNet)=@_;
        my @array=@$files;
        printf "Create Flow...\n";
        $flow = new Net::NfDump(
                        InputFiles => [@array],
                        Filter => 'src net '.$subNet,
                        Fields => 'srcip, flows',
                        Aggreg => 1, OrderBy => "flows"
                        );
        $flow->query();
        printf "Flow creation completed !\n";
        return $flow;
}
#Filling hash (keys : date) with specifics fields
sub flowsByIP {
        my ($files,$subNet,$h,$date)=@_;

        $flow =createFlow($files,$subNet);
        while (my ($srcip,$flows) = $flow->fetchrow_array() )  {
                if (((ip2txt($srcip)) eq "0.0.0.0")||
                ((ip2txt($srcip)) eq "0.0.0.0")){
                 $h->{$date}{ip2txt($srcip)} = $flows;
  }
        }
        $flow->finish();
}
#Return month length
sub monthLength{

        my $l;
        my($month)=@_;

        given($month){
                when ("01") {$l=31;}
                when ("02") {$l=28;}
                when ("03") {$l=31;}
                when ("04") {$l=30;}
                when ("05") {$l=31;}
                when ("06") {$l=30;}
                when ("07") {$l=31;}
                when ("08") {$l=31;}
                when ("09") {$l=30;}
                when ("10") {$l=31;}
                when ("11") {$l=30;}
                when ("12") {$l=31;}
```

```
        }

 return $l;
}
#Return list of nfcapd files for a specific date
sub GetFilesList
{
 my ($j,$m,$a) = @_;
 $strJou=sprintf("%02d",$j);
        $strMoi=sprintf("%02d",$m);
 my $Path ="/Volumes/VOL1/profiles-data/live/ucl/".$a."/".$strMoi."/".$strJou."/";
        my $FileFound;
        my @FilesList=();

        # Read files list
        opendir (my $FhRep, $Path) or die "Impossible d'ouvrir le repertoire $Path\n";
        my @Contenu = grep { !/^\.\.?$/ } readdir($FhRep);
        closedir ($FhRep);

        foreach my $FileFound (@Contenu) {
        # Files processing
         if ( -f "$Path/$FileFound"){

  push ( @FilesList, "$Path/$FileFound" );
         }
        # Directories processing
  elsif ( -d "$Path/$FileFound" ) {
  # Recursive search
   push (@FilesList, GetFilesList("$Path/$FileFound") );
                }
 }
 return sort @FilesList;
}
#Read hash, calculate totals and call drawing graphs functions
sub hashToData {

 my ($h,$subNet,$jour,$mois,$annee)=@_;
 $subNet=~tr/./_/;
        $subNet=(split(/\//,$subNet))[0];
 my $i = 0;
 my @data=();
 my @data2=();
    my @data3=();
 my $name;
 my %hash=%$h;
 my $tabNbr=1;
 foreach my $date (sort keys %hash) {
 $tabNbr=1;
 printf "Date : %s\n",$date;
                $data3[$i]=$date;
  foreach my $srcip (keys %{$hash{$date}}) {
   printf "%s -> %s\n",$srcip,$date;
   if ($srcip eq "0.0.0.0") {
              $data[$i]=$hash{$date}{$srcip};
   }
                        if ($srcip eq "0.0.0.0") {
                         $data2[$i]=$hash{$date}{$srcip};
   }
```

```
                $tabNbr++;
 }
$i++;
}
$name = $subNet."_".$mois."_".$annee;
graphMaker(\@data,\@data2,\@data3,$subNet,$jour."_".$mois."_".$annee);
}
#Return an array with days of a month
sub calcMois {

 my ($mois) =@_;

 my $nbr = monthLength($mois);

 my @jours;

 for (my $i = 1;$i<=$nbr;$i++){
   if ($i<10)
  {
   push @jours,"0".$i ;
  }
  else {
   push @jours,"".$i ;
  }
 }

return sort @jours;
}
#Draw graph
sub graphMaker {

my ($x,$y,$labels,$subNet,$date)=@_;

my $chart = Chart::Gnuplot->new(
        output => "daily_".$subNet."_".$date.".png",
        title  => "VLAN ".$subNet,
        xlabel => "Nombre de destinations",
        ylabel => "Volume (Kb)",
    );
my $dataSet = Chart::Gnuplot::DataSet->new(
        xdata => $x,
        ydata => $y,
        #title => "Plotting a line from Perl arrays",
        style => "points",
 pointtype =>'square',

);
my $j;
for (my $i=0;$i<=$#$labels;$i++){
 if ((@$y[$i])==0){
                        $j=1;
                }
                else {
                        $j=@$y[$i];
                }

 $chart->label (
  position => @$x[$i].",".$j." center",
```

```
        text    => @$labels[$i],
        offset => "0, -1,5",
        #pointcolor => 'dark-green',
        pointtype => 'square',
        #pointsize => 2,
 );
}
 $chart->set(
  logscale=>'xy',
  xrange => [1, 1000000],
  yrange => [1, 1000000000],
 );
 $chart->plot2d($dataSet);
}

printf "\\\\Analysing tool// \n";

#Array srcip|dstip|bytes
my %hash;
printf "Start analyzing...\n";
my @subnets = ('0.0.0.0/26');
my $mois = "06";
my $annee = "2015";
my @jours = calcMois($mois);
print Dumper \@jours;

foreach my $subNet(@subnets){
undef %hash;
 foreach my $jour(@jours) {
  printf "%d/%d/2013 for subnet : %s\n",$jour,$mois,$subNet;
  printf "-----------\n";
#Array for days of a month
  @fileList = GetFilesList($jour,$mois,$annee);

  foreach my $File  (@fileList) {
   print "$File\n";
  }
 my $files = \@fileList;
#Filling hash day per day
 flowsByIP($files,$subNet,\%hash,$jour);
 }
hashToData (\%hash,$subNet,$jour,$mois,$annee);
}
```

Code 6 – Script 4 - Evolution du nombre de flux

Annexe D : Script 5 - Analyse des fréquences des échanges entre deux hosts

```
#! /usr/bin/perl

use feature "switch";
use Net::NfDump qw ':all';
use Data::Dumper;
use Config::Simple;
use DBI;
use Text::CSV::Slurp;
use Text::CSV;
use Date::Business;
use Date::Business;
use Date::Format;
use POSIX qw(floor);
use Math::Trig;
use strict;
use warnings;
use GD::Graph::bars;
use GD::Graph::Data;


# functions

#Return list of nfcapd files for a specific date
sub getFilesList
{
        if (our $info == 1){
  printf "getFileList... \n";
 }
 my ($j,$m,$a) = @_;
        my $strJou=sprintf("%02d",$j);
        my $strMoi=sprintf("%02d",$m);
        my $Path ="/Users/cluycx/Desktop/Data/".$a."/".$strMoi."/".$strJou."/";
        my $FileFound;
        my @FilesList=();

        # Read files list
        opendir (my $FhRep, $Path) or die "Impossible d'ouvrir le repertoire $Path\n";
        my @Contenu = grep { !/^\.\.?$/ } readdir($FhRep);
        closedir ($FhRep);

        foreach my $FileFound (@Contenu) {
        # Files processing
                if ( -f "$Path/$FileFound") {
          push ( @FilesList, "$Path/$FileFound" );
                }
        # Directories processing
                elsif ( -d "$Path/$FileFound" ) {
                # Recursive search
                        push (@FilesList, GetFilesList("$Path/$FileFound") );
                }
        }
 if (our $info == 1){
```

```
                printf "getFilesList completed !\n";
        }
        return sort @FilesList;
}
#Flow creation (filtering on fields subnet,srcip, dstip,...))
sub createFlow {

 if (our $info == 1){
                printf "createFlow... \n";
        }

        my ($files,$subNet)=@_;
        my @array=@$files;
        my $flow = new Net::NfDump(
                        InputFiles => [@array],
                        Filter => 'src net '.$subNet,
                        Fields => 'srcip, dstip, dstport,srcport, first, last'
                        );
                        $flow->query();
 if (our $info == 1){
                printf "Flow creation completed !\n";
        }
                        return $flow;
}
#Filling hash (keys : scrip) with specifics fields
sub flowToHash {
my $i=0;
 if (our $info == 1){
                printf "flowToHash... \n";
        }
        my ($files,$subNet,$h,$flow,$ip)=@_;

        while (my ($srcip,$dstip,$dstport,$srcport,$first,$last) =
        $flow->fetchrow_array() ) {
  if ((ip2txt($srcip) eq $ip)&&((substr ip2txt($dstip), 0, 7) ne "130.104")
  &&((substr ip2txt($dstip), 0, 7) ne "192.168")) {
                 if (our $debug==1) {
    $i++;
   }
   $h->{ip2txt($srcip)}{ip2txt($dstip)}{$dstport}{$first}=$last;
  }
        }
        $flow->finish();
 if (our $info == 1){
                printf "Hash creation completed ! : %d\n",$i;
        }
}

#Pass 1 : minimum delay between same flow (scrip <=> dstip:port)
#Pass 2 : binary transformation (1 active, 0 inactive)
#Sampling frequency= min/4.

sub flowToBin {

my ($srcip,$dstip,$dstport,$first,$last)=@_;

  our $debut;
```

```
our $ecartTemp;
our $fin;
our $pass;
our @trace;
our $pos;
our $echanTime;
our $step;

              if (($first!=our $firstBack)&&($first>$firstBack)) {
                      if($debut==0) {
                             $debut = $first;
                      }
                      $fin = $last;
                      if ($ecartTemp == 0) {
                              $ecartTemp=$first+($last-$first);
                              if ($pass==2) {
   for (my $x = $echanTime+$step; $x<$first; $x+=$step) {
                                      $trace[$pos]=0;
                                      $pos++;
                                      $echanTime+=$step;
                               }

                               $trace[$pos]=1;
                               $pos++;
                               $echanTime+=$step;
                               for (my $j = $echanTime+$step; $j<$first+($last-
  $first); $j+=$step) {
                                      $trace[$pos]=1;
                                      $pos++;
                                      $echanTime+=$step;
                               }
  } #end pass 2
                      }
                      else {
                              if (our $ecartMin == 0){
                                     $ecartMin=$first - $ecartTemp;
                                     if ($pass==2) {
    for (my $k = $echanTime+$step; $k<$first; $k+=$step){
                                              $trace[$pos]=0;
                                              $pos++;
                                              $echanTime+=$step;
                                      }
   }#end pass 2
                              }
                              else {
                                      if ($ecartMin>$first-$ecartTemp) {
                                             $ecartMin=$first-$ecartTemp;
                                      }
   if ($pass==2) {
                                       for (my $l = $echanTime+$step; $l<$first;
   $l+=$step){
                                               $trace[$pos]=0;
                                              $pos++;
                                              $echanTime+=$step;
                                       }
   }#end pass2
                                      #$ecartTemp=$first+($last-$first);
                              }
```

```
                                $ecartTemp=$first+($last-$first);
    if ($pass==2) {
                                 $trace[$pos]=1;
                                 $pos++;
                                 $echanTime+=$step;
                                 for (my $m = $echanTime+$step; $m<$first+($last-
    $first); $m+=$step) {
                                         $trace[$pos]=1;
                                         $pos++;
                                         $echanTime+=$step;
                                 }#end for
                               }#end pass 2
                        } #end else

                        #backup $debut flux précédent (évite bug...)
                        $firstBack=$first;
                }#end if ((ip2txt($srcip) eq "0.0.0.0")&&(ip2txt($dstip)
                #eq "89.107.169.154")
                #&& ($dstport eq "80")&&($first!=$firstBack))
}
# Date to milliseconds
sub formatTime {
    my ( $milliseconds )  = @_;

    my $seconds = floor($milliseconds / 1000);
    my $msec = $milliseconds % 1000;

    my( $sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdst ) =
                                localtime( $seconds );
    $year += 1900;
    $mon += 1;

    return ( sprintf "%02d/%02d/%04d %02d:%02d:%02d:%04d",
                $mday,$mon,$year,$hour,$min,$sec,$msec );
}
# Milliseconds to en HH:MM:SS:milliseconds
sub formatTime2 {
    my ( $milliseconds )  = @_;

my $msec = $milliseconds % 1000;
my $x = $milliseconds / 1000;
my $sec = $x % 60;
$x /= 60;
my $min = $x % 60;
$x /= 60;
my $hour = $x % 24;

    return ( sprintf "%02d:%02d:%02d:%03d",$hour,$min,$sec,$msec );
}

# Fourier transform
#Call function to generate graph
sub fourier {

 my ($dstip,$dstport)=@_;
 my $name;
 my $nbr=0;
 my @data=();
```

```
 my $i = 0;
 our @trace;
 our $step;

 foreach my $x (@trace) {
   $nbr +=1;
 }

 printf "Nombre d'éléments : %d \n",$nbr;

 print Dumper(\@trace);

 for (my $k=1; $k<$nbr+1; $k+=1) {
  my $real = 0;
  my $img = 0;
         for (my $n=0; $n < $nbr ; $n++) {

                 $real += $trace[$n] * cos ((-2 * pi * $k * $n) / $nbr);
                 $img += $trace[$n] * sin ((-2 * pi * $k * $n) / $nbr);

         }
         my $result = sqrt(($real * $real) + ($img * $img));
  $data[0][$i] = ((int(((24*60*60*1000)/$step)/$nbr))*$k);
  $data[1][$i] = $result;
  $i++;
 }

 $name = $dstip."_".$dstport;
#Draw fraph frequency/amplitude
 graphMaker(\@data,$name);
}

#Rounding time
sub timeArrond {

        my ($t) = @_;
        my $newT;

        if ($t > 3600000) {
         $newT = (int(($t/3600000)+0.5))*3600000;
        }
        elsif ($t > 60000){
  $newT = (int(($t/60000)+0.5))*60000;
        }
        elsif ($t > 1000) {
         $newT = (int(($t/1000)+0.5))*1000;
        }
        else {
  $newT =  $t;
 }
        return $newT;
}

#Draw fraph frequency/amplitude
sub graphMaker {

 my ($data,$name) = @_;
```

```
 my $graph = GD::Graph::bars->new;
 my $title = (split(/_/,$name))[0].":".(split(/_/,$name))[1];
 $graph->set(
      x_label         => 'Frequences',
      y_label         => 'Amplitude',
      title           => $title,
      #y_max_value     => 7,
      #y_tick_number   => 8,
      #y_label_skip    => 3,
  #x_labels_vertical => 1,
  bar_width      => 1,
      #bar_spacing     => 10,
      #shadow_depth    => 4,
      #shadowclr       => 'dred',
  #transparent     => 0,
      correct_width  => 0,
 ) or die $graph->error;

 $graph->plot($data) or die $graph->error;
 $name =~tr/./_/;
 my $file = $name.".png";
 open(my $out, '>', $file) or die "Cannot open '$file' for write: $!";
 binmode $out;
 print $out $graph->gd->png;
 close $out;
}


#local values
my @subnets = ('0.0.0.0/26');
my $debJour = 01;
my $mois = 02;
my $annee = 2014;
my $nbrJours = 1;
my %hash;
my $ip = "0.0.0.0";


#global values
our $info = 1;
our $debug = 1;
our $firstBack;
our $debut;
our $fin;
our $ecartMin;
our $ecartTemp;
our @trace;
our $pos;
our $pass;


#1 fevrier 2014
our $echanTime=1391209200000;
our $beginDate = 1391209200000;
#1 janvier 2014
#$echanTime=1388530800000;
#$beginDate = 1388530800000;
```

```
#Sampling frequency
our $step;




#Main
printf « \\\\Analyse Fourier// \n";
printf "\n";

foreach my $subNet(@subnets){
 undef %hash;

        for (my $jour=$debJour;$jour<$debJour+$nbrJours;$jour+=1) {
                if ($info == 1) {
                        printf "%d/%d/%d for subnet : %s\n",$jour,$mois,$annee,$subNet;
                        printf "----------------------------------------\n";
                }
                #Tableau contenant liste fichiers d'un jour
                my @fileList = getFilesList($jour,$mois,$annee);

                if ($debug == 1) {
                        foreach my $File  (@fileList) {
                                print "$File\n";
                        }
                }

                #reference tableau contenant les liste des fichier ncapd
                my $files = \@fileList;

  #Remplissage tableau %hash jour par jour
         my $flow = createFlow($files,$subNet);

  flowToHash($files,$subNet,\%hash,$flow,$ip);
  if ($debug==1){
  }
  my $i;
  foreach my $srcip (keys %hash) {
   foreach my $dstip (keys %{$hash{$srcip} }) {
    foreach my $dstport (keys %{ $hash{$srcip}{$dstip} }){
    $pass=1;
    $step=0;
     while (($pass!=3)){
      $firstBack=0;
      $debut=0;
      $fin = 0;
      $ecartMin=0;
      $ecartTemp=0;
      $firstBack=0;
      @trace=();
      $pos=0;
      $echanTime=1391209200000;
      $beginDate = 1391209200000;
      foreach my $first (sort keys %{ $hash{$srcip}{$dstip}{$dstport} }){
       #printf "---------------> %s -> %s:%s first: %s last: %s\n", $srcip,
       $dstip,$dstport,formatTime($first),formatTime($hash{$srcip}{$dstip}{$dstport}
       {$first});
```

```
       flowToBin ($srcip,$dstip,$dstport,$first,$hash{$srcip}{$dstip}{$dstport}
       {$first});
      }
      if ($pass==2) {
       for (my $x = ($echanTime+$step); $x<=($beginDate+86400000); $x+=$step) {
                              #printf "%d->%s : 0\n",$x,formatTime2($x);
                              $trace[$pos]=0;
                              $pos++;
                              $echanTime+=$step;
                       }
      $pass=3;
      } #end pass 2
                     $beginDate=$beginDate+86400000;
                     if ($pass==1) {
       #If flow duration > 8 hours (keep)
       if (($fin-$debut)>28800000){
        $pass=2;
        printf "%s -> %s:%s Total duration : %s\n",$srcip,$dstip,$dstport,
        formatTime2($fin-$debut);
                       $step=timeArrond($ecartMin/4);
        printf"Echantillonage : %d->%s\n",$step,formatTime2($step);

                       }
       #If flow duration < 8 hours (delete)
       #if no ports remaining (delete source)
       else{
        $pass=3;
        delete($hash{$srcip}{$dstip}{$dstport});
        my $nbr=0;
        foreach my $dstport (sort keys %{ $hash{$srcip}{$dstip}}){
         $nbr++;
        }
        if ($nbr==0){
         delete($hash{$srcip}{$dstip});
        }

       }
      } #end pass 1
     }#end pass
     if ($step!=0){
                                                              printf "Step---->%s\n",$step;
                                                              fourier($dstip,$dstport);
                                                 }
    } #end foreach $dstport
   } #end foreach $dstip
  }#end foreach $srcip
  if ($debug==1){
  }
 }#end for jours

}#end foreach subnets
```

Code 7 – Script 5 - Analyse des fréquences des échanges entre deux hosts